IBM API Connect 10.0.7.0 CD

© Copyright IBM Corp. 2024.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Tables of Contents

API Connect overview 1
What's new in the latest release (Version 10.0.7.0) 2
What's new in Version 10.0.6.0 6
Language versions offered by API Connect 8
Known limitations 8
API Connect concepts 16
Who does what in API Connect? 16
API Connect user roles 17
Packaging strategy and terminology in API Connect 22
Understanding rate limits for APIs and Plans 25
API Connect components 26
Gateway types 28
API Connect support 29
API Connect glossary 29
Accessibility features for IBM API Connect 31
Legal information 32
Tracking API volume for auditing and compliance 32
Notices 32
Terms and conditions for information centers 34
IBM API Connect Considerations for GDPR Readiness 34
Essential reading 37

Tutorials 37

Installing and maintaining IBM API Connect 38
Kubernetes, OpenShift, and Cloud Pak for Integration 39
Planning your deployment 39
Prerequisite skills for administering API Connect 39
Key concepts: Custom Resources, Operators, and Operands 40
Deployment profiles, topologies, and strategies 40
Deployment and component profiles 40
API Connect deployment profiles for OpenShift and Cloud Pak for Integration 41
Component profiles, CPU limits, memory limits, and licensing 42
Management component deployment profile limits 42
Portal component deployment profile limits 49
Analytics component deployment profile limits 51
DataPower gateway deployment profile limits 52
Multiple data center deployment strategies 53
Two data center warm-standby deployment on Kubernetes and OpenShift 54
OpenShift: Deciding to use individual CRs, a top-level CR, or Cloud Pak for Integration 55
Network requirements for inter-subsystem communication 56
Enable JWT security instead of mTLS 57
In-cluster service communication between subsystems 58
Kubernetes ingress controller prerequisites 59
Persistent Volume (PV) requirements 60
Load balancer configuration in a Kubernetes deployment 60
Firewall requirements on Kubernetes 63
Planning your analytics deployment 65
Key Concepts: Cert-manager, Issuers, and Secrets 66
Certificates in a Kubernetes environment 68
Custom certificates on Kubernetes 72
Custom certificates reference 72
Generating custom certificates 74
Generating custom certificates using cert-manager in a two data center deployment 76
Configuring custom certificates before installation 78
Converting to custom front-end/ingress certificates after deployment 80
Webhooks deployed and managed by the API Connect Operator 80
Administering API Connect in your container environment 82
Namespaces in your container environment 82
Custom Resources and Custom Resource Definitions 82
Managing your API Connect CRs 84
Installing API Connect 84
Deployment requirements 84
IBM API Connect Version 10 software product compatibility requirements 87
API Connect licenses 88
Deployment procedures 88
Deploying on Kubernetes 88
Obtaining product files 88
Verifying the integrity of IBM product files 90
Deploying operators and cert-manager 91
Deploying operators in a single-namespace API Connect cluster 92
Deploying operators in a multi-namespace API Connect cluster 94

Installing cert-manager and certificates in a two data center deployment 96
Installing the API Connect subsystems 97
Installing the Management subsystem 98
Installing the DataPower Gateway subsystem 100
Installing the Developer Portal subsystem 103
Defining multiple portal endpoints for a Kubernetes environment 106
Installing the Analytics subsystem 107
Creating the analytics CR 107
Installing analytics 109
Installing a two data center deployment on Kubernetes 110
Converting a single data center to a two data center deployment on Kubernetes 113
Troubleshooting installation on Kubernetes 117
Deploying on OpenShift and Cloud Pak for Integration 118
Configuring FIPS support on OpenShift 118
Preparing for installation 121
Operator, CASE, and platform requirements 122
API Connect configuration settings 123
Basic configuration settings 123
Management subsystem settings 124
Gateway subsystem settings 127
Developer Portal subsystem settings 129
Analytics subsystem settings 130
Additional Kubernetes settings 130
Configuring timeouts for management endpoints 131
Installation procedures with IBM Cloud Pak for Integration 132
Installing API Connect in a single namespace on OpenShift 132
Installing operators 133
Setting up a certificate issuer 136
Installing the Management subsystem in a shared namespace 139
Installing the Gateway subsystem in a shared namespace 141
Installing the Portal subsystem in a shared namespace 145
Installing the Analytics subsystem in a shared namespace 148
Installing API Connect subsystems in different namespaces or environments on OpenShift 150
Installing operators 150
Installing the Management subsystem 153
Extracting the Management ingress-ca certificates 157
Installing the Gateway subsystem 158
Installing the Portal subsystem 162
Installing the Analytics subsystem 166
Installing with the top-level CR on OpenShift 169
Installing operators 169
Installing API Connect 172
Air-gapped installation 175
Installing with a bastion host 175
Installing with a portable computer or storage device 179
Installing a two data center deployment with Cloud Pak for Integration 184
Planning and initial preparation 184
Preparing your active data center 185
Preparing your warm-standby data center 187
Installing API Connect on the active data center 189
Installing API Connect on the warm-standby data center 190
Installing a two data center deployment on OpenShift 193
Planning and initial preparation 193
Preparing your active data center 194
Preparing your warm-standby data center 195
Installing API Connect on the active data center 200
Installing API Connect on the warm-standby data center 201
Enable additional features post-install 203
Enable management CA verification on REST API calls 203
Enable JWT security and disable mTLS between subsystems 204
Troubleshooting installation on OpenShift 205
Upgrading API Connect 206
Upgrading on Kubernetes 207
Planning your API Connect upgrade on Kubernetes 207
Pre-upgrade preparation and checks on Kubernetes 209
Upgrading API Connect on Kubernetes 211
Upgrading the management subsystem on Kubernetes 213
Upgrading the portal subsystem on Kubernetes 213
Upgrading the analytics subsystem on Kubernetes 214
Upgrading the gateway subsystem on Kubernetes 215
Troubleshooting upgrades on Kubernetes 216
Enabling gateway peering and verifying cluster status 217

Upgrading on OpenShift and Cloud Pak for Integration 218
Planning your API Connect upgrade on OpenShift 218
Operator, CASE, and platform requirements for upgrading 220
Pre-upgrade preparation and checks on OpenShift 221
Resolving duplicate users before upgrade on Cloud Pak for Integration 222
Pre-upgrade preparation for air-gapped environment using a bastion host 223
Pre-upgrade preparation for air-gapped environment using a portable storage device 227
Upgrading API Connect on OpenShift 232
Troubleshooting upgrade on OpenShift 235
Upgrading a two data center deployment on Kubernetes and OpenShift 240
Upgrading a two data center deployment on Cloud Pak for Integration 241
Maintaining API Connect 243
Advanced configuration 243
Installing License Service 244
Installing the Automated API behavior testing application 244
Installing the toolkit 246
Changing deployment profiles on Kubernetes and OpenShift 248
Changing deployment profiles on OpenShift top-level CR 249
Replica override for microservices 249
Changing the deployment profile on Cloud Pak for Integration 250
Enabling Developer Portal feature flags on Kubernetes 251
Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration 252
Enabling UI feature flags on Kubernetes 253
Enabling UI feature flags on OpenShift and Cloud Pak for Integration 254
Advanced configuration for the management subsystem 254
Updating management subsystem password, keys, and secrets 255
Changing the cloud administrator password 255
Changing the database encryption key 255
Changing the registration client_id and client_secret for applications 257
Configuring maximum size of client requests to the Management subsystem 259
Setting rate limits for public APIs on the management service for a Kubernetes environment 259
Enabling API governance on Kubernetes 260
Configuring monetization on Kubernetes 262
Overriding resources for Postgres components on Kubernetes V10.0.6.0 263
Overriding resources for Postgres components on OpenShift V10.0.6.0 264
Restarting management subsystem pods 265
Advanced configuration for the analytics subsystem 266
Advanced configuration for the gateway subsystem 266
Customizing a DataPower deployment 267
Updating the gateway password, secrets, and certificates 267
Changing the gateway administrator password 268
Changing the gateway client_secret 268
Changing the secret for gateway peering 269
Updating the gateway director and invocation endpoint profiles 269
Overriding the default DataPower image, version, and license 270
Enabling autoscaling of gateway pods 270
Configuring rate limits on the gateway nodes 271
Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment 271
Advanced configuration for the portal subsystem 272
Updating portal keys and secrets 272
Changing the secret for portal data and system tools 272
Changing the portal client_secret 274
Backing up, restoring, and disaster recovery 274
Backing up and restoring the management subsystem 275
Taking a management infrastructure configuration backup 275
Configuring management database backups 276
Running an on-demand backup of the management database 278
Restoring from a management database backup 279
Restoring the management database with a backup CR 279
Restoring the management database with a backup ID 280
Recovering from a disaster 281
Troubleshooting management database backup and restore 281
Backing up and restoring the developer portal 283
Taking a portal infrastructure configuration backup 283
Configuring portal database backups 284
Running an on-demand backup of the portal database 286
Managing your remote portal database backups 288
Restoring from a portal database backup 289
Recovering from a disaster 291
Troubleshooting portal database backup and restore 292
Backing up and restoring the analytics subsystem 293
Backing up and restoring on OpenShift top-level CR 293

Backing up your OpenShift top-level CR API Connect deployment 293
Disaster recovery on OpenShift top-level CR 293
Backing up and restoring on Cloud Pak for Integration and OpenShift top-level CR 294
Backing up your Cloud Pak for Integration API Connect deployment 294
Recovering from a disaster on Cloud Pak for Integration 295
Backing up, restoring, and disaster recovery V10.0.6.0 297
Backups on Kubernetes V10.0.6.0 298
Backing up and restoring the management subsystem v10.0.6.0 298
Configuring backup settings for fresh install of the Management subsystem v10.0.6.0 298
Configuring S3 backup settings for fresh install of the Management subsystem v10.0.6.0 299
Configuring SFTP backup settings for fresh install of the Management subsystem v10.0.6.0 301
Configuring local backup settings for fresh install of the Management subsystem v10.0.6.0 302
Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0 303
Generating a manual backup of the management subsystem v10.0.6.0 304
Restoring the management subsystem v10.0.6.0 305
Troubleshooting management subsystem backups v10.0.6.0 306
Troubleshooting resiliency issues v10.0.6.0 306
Troubleshooting stanza-create job for S3 backup configuration v10.0.6.0 307
Restore failure with bootstrap pod error v10.0.6.0 308
Restore failure with compliance pod error on Kubernetes v10.0.6.0 308
Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0 309
Overview of the Developer Portal backup resources v10.0.6.0 316
Backing up and restoring the analytics database 316
Backups on OpenShift V10.0.6.0 316
Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0 317
Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0 317
Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0 317
Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0 320
Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0 321
Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0 322
Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0 323
Restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0 325
Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration v10.0.6.0 326
Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration v10.0.6.0 326
Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration v10.0.6.0 327
Restore failure with compliance pod error on OpenShift and IBM Cloud Pak for Integration v10.0.6.0 328
Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0 328
Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0 329
Overview of the Developer Portal backup resources v10.0.6.0 316
Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0 332
Restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0 334
Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration v10.0.6.0 336
Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration 338
Disaster recovery V10.0.6.0 338
Disaster recovery on Kubernetes v10.0.6.0 338
Preparing for a disaster v10.0.6.0 338
Preparing the management subsystem for disaster recovery v10.0.6.0 339
Preparing the Developer Portal for disaster recovery v10.0.6.0 340
Preparing the Analytics subsystem for disaster recovery 342
Recovering from a disaster v10.0.6.0 342
Recovering the management subsystem on Kubernetes v10.0.6.0 342
Recovering the management subsystem from S3 backups v10.0.6.0 342
Recovering the management subsystem from SFTP backups v10.0.6.0 345
Recovering the Developer Portal after a disaster v10.0.6.0 347
Recovering the Analytics subsystem after a disaster 348
Disaster recovery on OpenShift and Cloud Pak for Integration v10.0.6.0 348
Preparing for disaster recovery on OpenShift v10.0.6.0 348
Preparing the Management subsystem for disaster recovery on OpenShift v10.0.6.0 349
Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0 350
Preparing the Analytics subsystem for disaster recovery on OpenShift 351
Recovering from a disaster on OpenShift v10.0.6.0 351
Preparing for disaster recovery on Cloud Pak for Integration v10.0.6.0 355
Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0 356
Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0 357
Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0 358
Recovering from a disaster on Cloud Pak for Integration v10.0.6.0 358
Maintaining a two data center deployment 363
Key concepts of 2DCDR and failure scenarios 363
Failing over to the warm-standby 365
Failing over to the warm-standby V10.0.6.0 368
Backup and restore requirements for a 2DCDR deployment 370
Removing a two data center deployment 371

Verifying replication between data centers 373
Synchronizing the ingress-ca certificates between sites 374
Troubleshooting a two data center deployment 375
Renewing TLS certificates 377
Renewing cert-manager controlled certificates 378
Renewing the management-ca with the ManagementSecretRotation CR 378
Renewing the portal-ca with the PortalSecretRotation CR 379
Renewing the analytics CA 380
Renewing the ingress-ca 381
Copying renewed ingress-ca to subsystems in different namespaces 381
Renewing end-entity certificates 382
Monitoring cert-manager certificate renewal 382
Renewing custom certificates 383
List of Issuers, CA certificates, and secrets 383
List of ingress certificates 384
List of intra-subsystem certificates 386
V2018 upgrade: List of certificates to update manually 387
Renewing certificates in a two data center deployment on Kubernetes and OpenShift 388
Monitoring and health checks 389
Monitoring on Kubernetes 389
Monitoring subsystems on Kubernetes 390
Monitoring Postgres disk usage 390
Monitoring platform health using Kubernetes events 391
Monitoring on OpenShift 392
Monitoring the API Connect cluster on OpenShift 392
Monitoring Postgres disk usage on OpenShift 393
Monitoring platform health on Red Hat OpenShift using Kubernetes events 395
Monitoring on Cloud Pak for Integration 395
Monitoring the API Connect cluster on Cloud Pak for Integration 395
Monitoring Postgres disk usage on Cloud Pak for Integration 397
Obtaining simple health check data of Developer Portal sites by using a REST API call 399
Logging 399
Gathering post-mortem logs 400
Changing logging levels 400
Troubleshooting 401
Troubleshooting the management subsystem on Kubernetes V10.0.6.0 401
Troubleshooting the management database V10.0.6.0 401
Increasing the PostgreSQL archive_timeout on Kubernetes V10.0.6.0 402
Recovering when disks are filled by the management database V10.0.6.0 403
Resizing a PersistentVolumeClaim for Postgres by volume expansion V10.0.6.0 403
Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC V10.0.6.0 404
Troubleshooting the management subsystem on OpenShift and Cloud Pak for Integration V10.0.6.0 406
Troubleshooting the management database on OpenShift and Cloud Pak for Integration V10.0.6.0 406
Increasing the PostgreSQL archive_timeout on OpenShift and Cloud Pak for Integration V10.0.6.0 407
Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database V10.0.6.0 408
Resizing a PVC for Postgres on OpenShift by using volume expansion V10.0.6.0 408
Resizing a PVC for Postgres on OpenShift by moving content V10.0.6.0 409
Power-cycling check on OpenShift and Cloud Pak for Integration 411
Uninstalling API Connect 411
Uninstalling API Connect subsystems 411
Removing Management subsystem postgres resources on Kubernetes V10.0.6.0 413
Removing API Connect 415
Removing postgres resources on Kubernetes after removing API Connect V10.0.6.0 417
Removing API Connect on OpenShift or Cloud Pak for Integration 418
Removing postgres resources on OpenShift and Cloud Pak for Integration V10.0.6.0 419
VMware 421
Planning your deployment 421
Prerequisite skills for administering API Connect on VMware 421
Deployment overview for endpoints 421
Planning your deployment topology and profiles 422
Enable JWT instead of mTLS 423
Firewall requirements on VMware 423
Firewall enabled ports for clustered OVA deployments 426
Load balancer ports for clustered OVA deployments 427
Load balancer configuration in a VMware deployment 427
Multiple data center deployment strategies 430
Two data center warm-standby deployment strategy on VMware 431
Planning your analytics deployment 432
API Connect TLS certificate best practices 432
Customizing your user-facing certificates 433
Installing API Connect 434
IBM API Connect Version 10 software product compatibility requirements 435

Requirements for initial deployment on VMware 435
API Connect licenses 437
What's new for v10 installation 438
First steps for deploying in a VMware environment 438
Verifying the integrity of IBM product files 440
Deploying the Management virtual appliance 441
Configuring the Management subsystem 441
Deploying the Management subsystem OVA file 447
Verify installation of the Management subsystem 448
Deploying the Analytics virtual appliance 449
Configuring the Analytics subsystem 450
Deploying the Analytics subsystem OVA file 453
Verifying deployment of the Analytics subsystem 454
Deploying the Developer Portal virtual appliance 455
Configuring the Developer Portal subsystem 456
Deploying the Developer Portal subsystem OVA file 460
Verifying deployment of the Developer Portal subsystem 461
Configuring two NICs on the Developer Portal 463
Specifying a range of allowable client IP addresses for Developer Portal 463
Defining multiple portal endpoints for a VMware environment 463
Deploying DataPower Gateway virtual appliance 466
Installing DataPower Gateway 466
Configuring DataPower Gateway for API Connect 466
Configuring DataPower API Gateway 467
Configuring DataPower Gateway (v5 compatible) 470
Sample configuration for multiple peering objects on gateway services external to Kubernetes 473
Post-deployment steps 477
Installing a two data center deployment 477
Upgrading API Connect 484
Planning your API Connect upgrade on VMware 485
Pre-upgrade preparation and checks VMware 486
Control Plane files for earlier releases 488
Checksum values for earlier releases 489
Upgrading management, portal, and analytics on VMware 489
Upgrading DataPower Gateway 492
Upgrading a two data center deployment 493
Management subsystem two data center upgrade from v10.0.6 494
Management subsystem two data center upgrade from v10.0.5 494
Troubleshooting upgrades on VMware 497
Troubleshooting upgrades to v10.0.6.0 on VMware 498
Maintaining API Connect 501
Advanced installation on VMware 501
Installing the IBM License Metric Tool on VMware 502
Installing the toolkit 246
Installing the Automated API behavior testing application 504
Advanced configuration on VMware 505
Adding a static route on a virtual machine 505
Adding disk space to a VMware appliance 506
Appliance boot mode configuration 506
Running commands at boot with iso-ignored boot mode 508
Changing deployment profiles on VMware 509
Changing the DNS server configuration on appliances 510
Configuring API Connect subsystems in a cluster 510
Configuring SSHD to limit access to your deployment 514
Enabling Developer Portal feature flags on VMware 515
Managing an appliance data disk 516
Replica override for microservices on VMware 516
Testing a new configuration against a current cluster 516
TLS certificate concepts and operations 517
Certificate reference VMware 518
Customizing user-facing certificates 523
Certificate expiry and renewal 523
Extracting and copying certificates 525
Rotating the portal intra-subsystem certificates on VMware 525
Apicup certs command reference 526
Certificate validation reference 527
Use JWT security instead of mTLS between subsystems 528
Credentials and encryption secrets reference 528
Advanced configuration for management subsystem 529
Setting rate limits for public APIs on the management service 529
Configuring maximum size of client requests to the Management subsystem on VMware 531
Enabling API governance on VMware 532

Configuring monetization on VMware 533
Configuring use of an external NTP server 534
Overriding resources for Postgres components on VMware V10.0.6.0 535
Enabling UI feature flags on VMware 536
Configuring rate limits on the gateway nodes on VMware 537
Backing up and restoring on VMware 537
Backing up and restoring the Management subsystem 538
Configuring management database backups 538
Running an on-demand backup of the management database 540
Restoring the management database backup 541
Recovering from a disaster 542
Backing up and restoring the Management subsystem V10.0.6.0 542
Configuring backup settings during initial installation of the management subsystem V10.0.6.0 543
Verify configuration for s3 backup V10.0.6.0 545
Reconfiguring backup settings for the management subsystem V10.0.6.0 547
Backing up the management subsystem V10.0.6.0 549
Restoring the management subsystem V10.0.6.0 550
Troubleshooting management subsystem backups on VMware V10.0.6.0 551
Troubleshooting resiliency issues V10.0.6.0 551
Troubleshooting stanza-create job for S3 backup configuration V10.0.6.0 552
Restore failure with compliance pod error on VMware V10.0.6.0 553
Backing up and restoring the Developer Portal 553
Configuring backup settings for the Developer Portal subsystem 554
Overview of the Developer Portal backup resources v10.0.6.0 316
Backing up the Developer Portal subsystem 557
Restoring the Developer Portal subsystem 558
Backing up and restoring the Analytics database on VMware 559
Configuring backup settings for Analytics 560
Running a backup of the Analytics database 561
Restoring the Analytics database 562
Disaster recovery 563
Preparing for a disaster 563
Preparing the management subsystem for disaster recovery on VMware V10.0.6.0 563
Preparing the Developer Portal subsystem for disaster recovery on VMware 564
Preparing the Analytics subsystem for disaster recovery on VMware 564
Recovering from a disaster 565
Recovering the Management subsystem on VMware V10.0.6.0 565
Recovering the Developer Portal subsystem on VMware 567
Recovering the Analytics subsystem on VMware 568
Using VM snapshots for infrastructure backup and disaster recovery 568
Disabling and re-enabling the analytics subsystem on VMware 570
Maintaining a two data center deployment 570
Backup and restore considerations for a two data center deployment 570
How to failover a two data center deployment 571
Failover steps when active site is responsive 571
Failover steps when active site is inaccessible 573
How to failover a two data center deployment V10.0.6.0 574
Recovering from a failover of a two data center deployment V10.0.6.0 578
Removing a two data center deployment 580
Monitoring and health checks 581
Checking cluster health on VMware 581
Monitoring Postgres disk usage on VMware 582
Obtaining simple health check data of Developer Portal sites by using a REST API call 583
Monitoring the network with SNMP 583
Logging 584
Configuring remote logging for a VMware deployment 584
Gathering logs for a VMware environment 586
Changing logging levels 587
Troubleshooting 588
Recreating a calico pod that is in the CrashLoopBackOff state 588
Running a filesystem check on a VMware root partition 588
Troubleshooting the management database V10.0.6.0 589
Increasing the PostgreSQL archive_timeout on VMware 590
Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment 590
Troubleshooting analytics on VMware 591
Migrating from IBM API Connect v5 Public Cloud to v10.0.6x 591
Migrating a version 5 deployment to version 10.0.x 591
Migrating v5-compatible APIs to API Gateway 591
Preparing the environment 592
Planning your API migration 592
Downloading and extracting the data to migrate 593
Converting v5-compatible API definitions 593

Pushing custom policies and gateway extensions to the DataPower API Gateway 598
Pushing provider organization data 600
Migrating from v10 to v10 on a different form factor 604
v10 migration overview 605
v10 migration requirements 607
v10 migration steps 608
Preparing the source system 609
Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration 610
Installing the target system on VMware 613
Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration 614
Restoring the target system on VMware 619
Verifying the target system on Kubernetes, OpenShift, and Cloud Pak for Integration 620
Verifying the target system on VMware 621
Migrating from OpenShift to Cloud Pak for Integration on the same cluster 621
v10 migration troubleshooting 623
Shutting down the Postgres database 624
[Technical Preview] Using Config Sync to replicate consumer-side catalog data 626
Downloading the API Connect Config Sync utility 627
Setting environment variables for Config Sync 628
Running Config Sync as a standalone binary 630
Running Config Sync as a cronjob on OpenShift 630
Running Config Sync as a cronjob on Kubernetes 631
Running Config Sync as a cronjob on VMware 632
The API Connect operations tool: apicops 633

Configuring and managing your server environment 633
Cloud Manager configuration checklist 634
Activating your Cloud Manager user account 635
Accessing the Cloud Manager user interface 636
Defining your Cloud Manager topology 637
Creating an availability zone 638
Registering a gateway service 639
Registering an analytics service 641
Registering a portal service 643
Associating an analytics service with a gateway service 644
Setting visibility for a service 645
Managing authentication and security 646
User registries overview 646
Configuring an Authentication URL user registry 647
Configuring a shared custom user registry 649
Configuring an LDAP user registry in the Cloud Manager 654
Using the CLI to configure a shared LDAP user registry 656
Configuring a Local User Registry 663
Configuring an OIDC user registry in Cloud Manager 664
Setting visibility for a user registry 670
Deleting a user registry 670
Removing a user from a user registry 670
Removing yourself from a user registry 671
Removing another user from a user registry 671
TLS profiles overview 672
Viewing TLS Profiles, Keystores, and Truststores 673
Creating a TLS Server Profile 674
Creating a TLS Client Profile 675
Setting visibility for a TLS Client Profile 676
Defining elliptic curve cryptographic schemes for a TLS client profile 676
Creating a Keystore 677
Creating a Truststore 678
Viewing certificate details and adding certificates to a keystore or truststore 679
Generating a self-signed certificate using OpenSSL 679
Generating a PKCS#12 file for Certificate Authority 680
Setting the API invocation certificate for a gateway service 681
Updating the PKCS#12 certificate for a TLS server profile 682
OAuth Provider overview 682
Configuring a native OAuth provider 683
Configuring basic settings for a native OAuth provider 685
Configuring scopes for a native OAuth provider 685
Configuring user security for a native OAuth provider 686
Configuring tokens for a native OAuth provider 687
Configuring token management and revocation for a native OAuth provider 688
Configuring introspection for a native OAuth provider 689
Configuring metadata for a native OAuth provider 690
Configuring the OIDC parameters for a native OAuth provider 690
Editing a native OAuth provider by using the API Editor 691
Configuring a third-party OAuth provider 692
Setting the visibility for OAuth providers 693

OAuth concepts for API Connect 694
OAuth user scenario 694
OAuth introspection for third-party OAuth providers 695
Oauth metadata URL and authentication URL 696
Scope 699
Tokens 703
Refresh tokens 703
OAuth revocation URL 703
Authenticating and authorizing through a redirect URL 705
Token management with the DataPower Gateway (v5 compatible) 707
Token management with the DataPower API Gateway 709
Authentication URL 712
Custom forms for user security 713
Creating a custom HTML login form for user security 713
Creating a custom HTML authorization form for user security 714
Securing an API with a JSON Web Token 715
Troubleshooting OAuth 715
Configuring the cloud settings 715
Change the name of your cloud 716
Configuring an email server for notifications 717
Setting up notifications 718
Customizing email notification templates 718
Configuring invitation timeouts 720
Configuring API key settings 721
Using the CLI to modify cloud settings 721
Configuring Base64 encoding for temporary tokens 722
Configuring the password-reset notification and timeout 723
Selecting user registries for Cloud Manager and API Manager 723
Setting a default user registry for Cloud Manager and API Manager 724
Configuring timeouts for access tokens and refresh tokens 724
Viewing platform and UI endpoints 725
Configuring the default gateway services for Catalogs 726
Managing the public/private key pairs for signing tokens 726
Configuring public/private key pairs for tokens 727
Configuring API governance in the Cloud Manager 729
The API governance ruleset lifecycle in the Cloud Manager 731
Updating rulesets and rules with the toolkit CLI 732
Administering provider organizations 733
Creating a provider organization 733
Editing a provider organization 734
Deleting a provider organization 735
Viewing current provider organizations 736
Sending messages to provider organization owners 736
Administering members and roles 737
Creating roles in the admin organization 737
Editing roles in the admin organization 738
Configuring LDAP group mappings on Cloud Manager user roles 739
Deleting roles in the admin organization 741
Viewing members and roles 741
Adding members to the admin organization 741
Assigning roles to members 742
Deleting a member 743
Role Defaults overview 743
Managing role defaults for provider organizations 744
Managing role defaults for consumer organizations 744
Monitoring the cloud 745
Accessing and configuring analytics 745
Configuring audit logging to monitor user operations 745
Configuring the audit settings 748
Reviewing a gateway's processing status 751
Changing your Cloud Manager password and profile information 753
Resolving login problems by increasing HTTP header size 754
Onboarding a new admin for Cloud Pak for Integration 755
Extending the Gateway server behavior 756
Gateway extension guidelines - DataPower Gateway (v5 compatible) 756
Gateway extension guidelines - DataPower API Gateway 757
Gateway extensions manifest 760
GWD extension type 762
config-sequence 763
parse-settings 763
analytics-endpoint 763
front-side-handler 763
apigw 764
tls-client 764
api-collection 765
api-plan 766

registry 768
oauth-provider 769
api-definition 770
assembly 774
v5 policy emulation limitations 779
Configuring your Gateway server extensions 779
Cloud Manager Tutorials 781
Tutorial: Configuring the Cloud 781
Tutorial: Creating a Provider Organization 789

Developing your APIs and applications 792
API development checklist 793
Working with the toolkit 793
Installing the toolkit 246
Logging in from API Designer 796
Setting the locale for API Designer 797
Switching clouds from API Designer 798
Working offline with API Designer 798
Gathering support information in API Designer 798
Using the developer toolkit command-line tool 799
Overview of the command-line tool 800
Logging in to a management server 802
Logging in to a management server with an OIDC registry 804
Cloud administration commands 805
API development and management commands 808
Creating APIs and applications 814
Publishing APIs and applications 815
Managing API Products 816
Working with Drafts 819
Reading input from the command line 819
Scripting with the toolkit commands 821
Working with OpenAPI extensions 822
Using the toolkit CLI to execute API tests 826
Test definition syntax 827
Variable assignment in YAML configuration 829
Managing platform REST API keys 830
Searching for items in API Manager 830
Working with API definitions 830
Creating an API definition 831
Creating a REST proxy API from a target service 832
Creating a REST proxy API from an existing OpenAPI service 833
Creating a REST proxy API from an existing WSDL service 835
Creating a SOAP proxy API from an existing WSDL service 836
Creating a new REST OpenAPI definition 838
Adding a REST API by importing an OpenAPI definition file 839
Adding a SOAP API by importing a zip file 840
Creating a GraphQL proxy API 841
Using the GraphQL schema editor 843
Supporting introspection for a GraphQL API 846
Enabling the GraphiQL editor for a GraphQL API 846
Enabling the cost endpoint for a GraphQL API 847
Securing a GraphQL API by using a client ID 847
Request mechanisms supported by GraphQL endpoints 849
GraphQL custom schema directives and scalar types 850
@remove directive 850
GraphQL limitations 851
Importing an API from the CP4I asset repository 851
Editing an OpenAPI 2.0 API definition 852
Specifying API metadata 853
Specifying the host for an API 853
Specifying the basepath for an API 854
Specifying the schemes for an API 854
Specifying the MIME types that an API consumes 854
Specifying the MIME types that an API produces 855
Enforcing security requirements on an API 855
Defining security schemes 856
Defining basic authentication security schemes 856
Defining API key security schemes 857
Client ID behavior in API key security definitions 858
Defining OAuth2 security schemes 859
Enabling CORS support for an API 860
Specifying external documentation for an API 861
Adding tags to an API 861

Defining Paths for an API 862
Defining parameters for a path 863
Defining operations for a path 863
Adding tags to an operation 864
Specifying external documentation for an operation 864
Specifying the MIME types that an API operation produces 864
Specifying the MIME types that an API operation consumes 865
Defining parameters for an operation 865
Defining responses for an operation 865
Specifying schemes for an operation 866
Enforcing security requirements on an operation 866
Defining schema definitions for an API 867
Creating schema definitions 867
Editing schema definitions 868
Defining parameters for an API 869
Creating a parameter 870
Editing a parameter 870
Defining responses for an API 871
Creating a response 871
Editing a response 872
Specifying gateway and portal settings 872
Defining target services for an API 873
Setting API properties 873
Defining Catalog specific property values 874
Configuring application authentication for an API 874
Configuring activity logging 875
Configuring v5 compatibility options 876
Including elements in your API assembly 876
The assembly editor 877
Adding elements to your assembly 878
Handling errors in the assembly 879
Specifying the gateway type for an API definition 879
Converting an API definition for deployment to the DataPower API Gateway 880
Modifying a GraphQL schema 880
Validating the OpenAPI YAML source 881
Validating an API document by using API governance 881
Validating an API document by using the toolkit CLI 883
Editing an OpenAPI 3.0 API definition 884
OpenAPI 3.0 support in IBM API Connect 885
Specifying API metadata 885
Defining servers for an API 886
Enforcing security requirements on an API 887
Enabling CORS support for an API 888
Specifying external documentation for an API 889
Adding tags to an API 889
Defining Paths for an API 890
Defining servers for a Path 891
Defining parameters for a Path 891
Defining operations for a Path 892
Adding tags to an operation 892
Specifying external documentation for an operation 893
Defining parameters for an operation 893
Defining the request body for an operation 894
Defining responses for an operation 894
Enforcing security requirements on an operation 895
Defining servers for an operation 896
Defining components for an API 896
Defining schema components 897
Creating a schema component 897
Editing a schema component 898
Defining response components 900
Creating a response 900
Editing a response 901
Creating a content definition 902
Editing a content definition 902
Creating an encoding definition 903
Editing an encoding definition 903
Defining parameter components 904
Creating a parameter 905
Editing a parameter 906
Defining example components 907
Creating an example 907

Editing an example 908
Defining request body components 908
Creating a request body 909
Editing a request body 909
Defining header components 910
Creating a header 910
Editing a header 911
Defining security scheme components 912
Defining basic authentication security scheme components 912
Defining API key security scheme components 914
Client ID behavior in API key security definitions 858
Defining OAuth2 security scheme components 915
Defining an HTTP bearer security scheme 917
Defining link components 917
Creating a link 918
Editing a link 918
Specifying gateway and portal settings 919
Defining target services for an API 920
Setting API properties 920
Defining Catalog specific property values 921
Configuring application authentication for an API 922
Configuring activity logging 922
Including elements in your API assembly 923
The assembly editor 923
Adding elements to your assembly 925
Handling errors in the assembly 925
Validating the OpenAPI YAML source 881
Validating an API document by using API governance 881
Validating an API document by using the toolkit CLI 883
LDAP authentication 929
Authentication URL user registry 930
Activating an API 931
Testing an API 932
Locating API information on the Endpoints tab 933
Using the Test tab to debug your API 933
Preparing an API for debugging with the Test tab 933
Specifying the testing preferences for an API 934
Sending the API request 935
Reviewing the API response and trace 936
Testing a GraphQL API with the Test tab 937
Testing an API with Automated API behavior testing 937
Getting started 938
Creating a test suite 938
Modifying a test suite 939
Deleting a test suite 940
Creating a test case 941
Modifying a test case 943
Deleting a test case 944
Defining variables 944
Running a test case 946
Viewing a test report 946
Publishing a test case 948
Tagging a test 948
API Hooks 949
Creating an API Hook 949
Creating An API Keys and Secrets 950
Using API Hooks 951
List All Tests 951
Run All Published Tests 952
Run Published Tests By Tag 955
Run Published Test By ID 957
Test definition syntax 960
Invoking test cases from Jenkins with JUnit result format 962
AutoTest Assist 965
Creating a profile 966
Modifying a profile 967
API extensions 967
Deleting a profile 969
Running a profile 969
Viewing the report 970
Profile configuration 973
Profile security 978

Data generation rules 978
Testing an API with the Policies editor 982
Testing an API with the Local Test Environment 983
Testing an API with the Explorer tab 986
Staging an API 987
Publishing an API 988
Creating a new version of an API definition 990
Updating an API definition from a file 990
Updating a SOAP API 991
Renaming an API definition 991
Changing an API rate limit 992
Adding an OpenAPI extension to an API 992
Downloading an API definition 993
Deleting an API definition 993
Using an options file when importing a WSDL service 994
Variable references in API Connect 994
API properties 997
API policies and logic constructs 1001
Built-in policies 1002
Activity Log 1004
Client Security 1005
Extract 1006
Constructing JSONata expressions to extract and transform data 1007
GatewayScript 1009
GatewayScript code examples 1010
Generate JWT 1012
Validate JWT 1014
GraphQL introspect 1016
Invoke 1016
Configuring the Invoke policy for DataPower API Gateway 1017
Configuring the Invoke policy for DataPower Gateway (v5 compatible) 1020
JSON to XML 1022
Configuring the JSON to XML policy for DataPower API Gateway 1022
Configuring the JSON to XML policy for DataPower Gateway (v5 compatible) 1023
Log 1024
Map 1025
The Map policy structure 1026
Configuring the Map policy in the user interface 1029
Map policy examples 1032
OAuth 1035
Example - using multiple OAuth policies in an OAuth provider assembly 1037
Parse 1040
Proxy 1042
Rate Limit 1045
Configuring a rate, burst, or count limit on the DataPower API Gateway 1046
Redaction - DataPower API Gateway 1048
Constructing JSONata expressions to redact fields 1048
Redaction - DataPower Gateway (v5 compatible) 1051
Constructing XPath expressions to redact fields 1052
Set Variable 1054
Configuring the Set Variable policy for DataPower API Gateway 1055
Configuring the Set Variable policy for DataPower Gateway (v5 compatible) 1056
User Security 1057
Validate - DataPower API Gateway 1059
Validate - DataPower Gateway (v5 compatible) 1062
Validate Username Token 1063
Websocket Upgrade 1064
Configuring the Websocket Upgrade policy for DataPower API Gateway 1064
Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible) 1066
XML to JSON 1067
Configuring the XML to JSON policy for DataPower API Gateway 1068
Configuring the XML to JSON policy for DataPower Gateway (v5 compatible) 1069
XSLT 1069
Configuring the XSLT policy for DataPower API Gateway 1070
Configuring the XSLT policy for DataPower Gateway (v5 compatible) 1072
XSLT policy examples 1073
Implementation code examples 1076
Logic Constructs 1081
if 1082
operation-switch 1082
switch 1083
Using the switch policy condition editor 1083
throw 1087

Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway 1088
Manipulating attachments in a message object 1091
User-defined policies 1092
Tags in API Connect 1092
Working with Products 1093
Creating a draft Product 1094
Downloading a draft Product 1096
Importing a draft Product 1096
Searching for a draft Product 1097
Editing a draft Product 1098
Defining rate limits for an API operation 1101
Staging a draft Product 1102
Publishing a draft Product 1103
Creating a new version of your Product 1105
Specifying the gateway type for a Product 1106
Deleting a draft Product 1106
Organizing your Products into categories 1107
Creating and validating API and Product definitions by using the CLI 1107
Creating an OpenAPI definition file 1108
IBM extensions to the OpenAPI specification 1108
execute 1110
activity-log 1111
client-security 1112
extract 1113
gatewayscript 1114
graphql-introspect 1115
if 1116
invoke 1117
json-to-xml 1121
jwt-generate 1121
jwt-validate 1123
log 1124
map 1125
operation-switch 1130
oauth 1131
Example - using multiple OAuth policies in an OAuth provider assembly 1037
parse 1136
proxy 1139
ratelimit 1140
redact - DataPower API Gateway 1142
redact - DataPower Gateway (v5 compatible) 1143
set-variable 1145
switch 1146
Writing switch condition scripts - DataPower API Gateway 1147
Writing switch condition scripts - DataPower Gateway (v5 compatible) 1150
throw 1151
User-defined policies 1152
user-security 1152
validate - DataPower API Gateway 1154
validate - DataPower Gateway (v5 compatible) 1157
validate-usernametoken 1158
websocket-upgrade 1159
xml-to-json 1162
xslt 1162
catch 1165
properties 1165
catalogs 1166
activity-log 1166
Specifying a gateway type for an API definition 1167
Referring to an extension in an API definition 1168
Using $ref to reuse code fragments in your OpenAPI files 1169
Creating a Product definition file 1170
Product definition schema 1171
Completing the information section of your Product description 1179
Specifying a gateway type for your Product 1180
Converting a Product YAML file to use DataPower API Gateway 1181
Specifying the visibility of your Product 1181
Referencing the APIs for your Product 1183
Describing Plans in your Product 1183
An example YAML representation of a Product 1186
Using x-ibm-languages to create multilingual API and Product documentation 1186
Using x-example to control the examples displayed in the Developer Portal 1189
Using x-embedded-doc to add additional documentation to products and APIs 1190
Using x-pathalias to give consistent URLs for products and APIs 1193

Validating the YAML or JSON definition of an API or Product 1194
Creating and using API and Product definitions templates 1194
Working with global policies 1194
Customizing the preflow policies 1198
Reference 1199
API Connect context variables 1199
OAuth context variables 1204
Error cases supported by assembly catches 1207
API and Product definition template examples 1207
Template variables for API and Product definitions 1214

Managing your APIs 1215
API management checklist 1216
Activating your API Manager user account 1217
Accessing the API Manager user interface 1218
Searching for items in API Manager 1218
Working with Catalogs 1219
Creating and configuring catalogs 1219
Configuring Product visibility at a Catalog or Space level 1224
Configuring sub paths for Developer Portal sites 1226
Managing Catalog membership 1226
Importing a user-defined policy into a Catalog 1227
Importing an OpenAPI extension into a Catalog 1227
Retaining Version 5 vanity endpoint behavior in a Catalog 1228
Managing the application lifecycle 1229
Approving application upgrade requests 1229
Using syndication in API Connect 1230
Enabling Spaces in a Catalog 1230
Creating, modifying, and deleting Spaces 1231
Working with Spaces 1232
Managing Products in a Space 1232
Managing subscription requests in a Space 1233
Managing application subscriptions in a Space 1233
Managing Space membership 1233
Managing user access in a Space 1234
Managing Gateways in a Space 1234
Working with Products 1235
The Product lifecycle 1235
Monetizing your Products 1237
Adding a billing integration resource 1238
Billing integration resource job queues 1238
Adding a billing integration resource to a Catalog 1239
Defining a Product with billing integration 1240
Managing your Products 1241
Considerations when changing a Product lifecycle with billing integration 1242
Publishing a Product 1243
Publishing a new Product 1243
Replacing a Product with another Product 1244
Superseding a Product with another Product 1245
Migrating application subscriptions to another Product 1247
Migrating application subscribers to new Product versions 1247
Changing the availability of a Product 1248
Deprecating a Product 1249
Retiring a Product 1250
Re-staging a Product 1250
Working with the subscriptions in a Product 1251
Updating the gateway services for a Product 1252
Removing a Product from a Catalog 1252
Approving Product lifecycle and subscription requests 1253
Working with APIs 1254
Working with Plans 1254
Working with consumer organizations 1255
Creating a consumer organization 1255
Editing a consumer organization 1256
Deleting a consumer organization 1257
Working with the applications in a consumer organization 1258
Working with the subscriptions in a consumer organization 1258
Working with consumer organization groups 1258
Sending messages to consumer organization owners 1259
Working with developer applications 1260
Working with application subscriptions 1260
Security and authentication 1261
Creating a TLS client profile 1261
Creating a Keystore 1262
Creating a Truststore 1263
Generating a PKCS#12 file for Certificate Authority 1264
Generating a self-signed certificate using OpenSSL 1265

Viewing certificate details and adding certificates to a keystore or truststore 1265
Defining elliptic curve cryptographic schemes for a TLS client profile 1266
Authenticating by using your enterprise user registry 1267
Working with user registries 1267
Creating an Authentication URL user registry 1268
Creating an organization-specific custom user registry 1269
Creating an LDAP user registry in API Manager 1272
Using the CLI to create an organization-specific LDAP user registry 1275
Creating a Local User Registry 1281
Creating an OIDC user registry in API Manager 1281
Modifying the configuration details for a user registry 1287
Password lockout criteria 1288
Configuring a native OAuth provider 1288
Configuring basic settings for a native OAuth provider 1290
Configuring scopes for a native OAuth provider 1291
Configuring user security for a native OAuth provider 1291
Configuring tokens for a native OAuth provider 1292
Configuring token management and revocation for a native OAuth provider 1293
Configuring introspection for a native OAuth provider 1294
Configuring metadata for a native OAuth provider 1295
Configuring the OIDC parameters for a native OAuth provider 1295
Editing a native OAuth provider by using the API Editor 1296
Configuring a third-party OAuth provider 1296
OAuth concepts for API Connect 694
OAuth user scenario 694
OAuth introspection for third-party OAuth providers 695
OAuth external URL and authentication URL 696
Scope 699
Tokens 703
Refresh tokens 703
Token revocation 703
Authenticating and authorizing through a redirect URL 705
Token management with the DataPower Gateway (v5 compatible) 707
Token management with the DataPower API Gateway 709
Authentication URL user registry 712
Custom forms for user security 713
Creating a custom HTML login form for user security 713
Creating a custom HTML authorization form for user security 714
Securing an API with a JSON Web Token 715
Troubleshooting OAuth 715
API Analytics 1320
Catalogs, spaces, and analytics 1320
Accessing analytics 1320
Configuring API governance in the API Manager 1321
The API governance ruleset lifecycle in the API Manager 1323
Updating rulesets and rules with the toolkit CLI 1324
Administering user access 1325
Setting the expiration for invitations 1325
Configuring notifications 1326
Configuring sender details for email notifications 1327
Viewing your user information 1330
Adding provider organization users and assigning roles 1330
Creating custom roles 1331
Removing a user from a provider organization 1331
Configuring LDAP group mappings on API Manager user roles 1332
Changing your API Manager password 1334
Resolving login problems by increasing HTTP header size 1334
Reviewing a gateway's processing status 1335
Reference 1336
API gateway response codes 1336
Troubleshooting your billing configuration 1336
Authoring policies 1337
Authoring policies for the DataPower Gateway (v5 compatible) 1337
Describing your policy 1338
Creating a new user-defined policy 1341
Implementing your policy 1341
Packaging and importing your policies into IBM API Connect 1343
Reference 1344
Implementation code examples 1344
Authoring policies for the DataPower API Gateway 1348
Defining, packaging, and publishing a catalog-scoped policy for the API Gateway 1349
Structure of the catalog-scoped user-defined policy YAML file 1351
Defining, packaging, and publishing a global-scoped policy for the API Gateway 1353
Developer Portal: socialize your APIs 1355

Using the Developer Portal 1355
Logging in as a consumer to the Developer Portal by using the CLI 1356
Exploring APIs and Products in the Developer Portal 1356
How to use a URL to navigate directly to a Product or API in the Developer Portal 1357
Testing an API by using the Developer Portal test tool 1357
Calling an API 1358
Calling an API by using CORS 1359
Applications in the Developer Portal 1360
Registering an application 1360
Upgrading a Development application to Production status 1361
Managing applications 1362
Editing an application 1362
Deleting an application 1362
Changing an application image 1363
Verifying an application client secret 1363
Analytics in the Developer Portal 1363
Application analytics in the Developer Portal 1363
Organization analytics in the Developer Portal 1363
Consumer organizations in the Developer Portal 1364
Adding a Consumer organization from within the Developer Portal 1364
Editing the name of a Consumer organization 1364
Switching Consumer organizations 1365
Adding users to a Consumer organization 1365
Removing a user from a Consumer organization 1365
Changing the ownership of a Consumer organization 1366
Deleting a Consumer organization 1367
User accounts, passwords, and support in the Developer Portal 1368
Creating a new developer account 1369
Changing your account settings 1369
Deleting your developer account 1369
Configuring the Developer Portal site 1370
Concepts in the Developer Portal 1371
Getting started configuring the Developer Portal site 1372
Appearance 1374
Controlling general appearance elements in the Developer Portal 1374
Changing the shortcut icon 1374
Changing the site email address 1375
Changing the site logo 1375
Changing the site name or site slogan 1375
Changing the visibility of site branding 1376
Adding custom JavaScript to a custom theme 1376
Applying a modified content type template 1376
Creating a sub-theme 1377
Installing additional themes 1380
Deleting themes 1381
Content 1381
Adding content elements in the Developer Portal 1381
Adding and configuring meta tags 1382
Disabling external search engine indexing 1383
Adding custom pages 1383
Integrating X data into the social block 1383
Editing the social block 1384
Adding content in multiple languages 1384
Adding custom pages to APIs and Products 1385
Adding new frequently asked questions 1385
Applying an image to an API or Product 1386
Attaching a documentation file to APIs 1386
Configuring blogs 1387
Configuring the taxonomy menu block 1387
Customizing the URL alias for a specific API or Product page 1388
Embedding multimedia in site content 1388
Linking from one piece of site content to another 1388
Linking to social media sites 1389
Managing tags in the Developer Portal 1389
Tagging APIs based on their lifecycle phase 1390
Posting a poll 1390
Adding images to site content 1390
Configuring and restricting content in the Developer Portal 1391
Creating content types 1391
Adding fields to content types 1392
Configuring documentation upload limitations 1392
Configuring image upload limitations 1393
Configuring the appearance of ratings in the Developer Portal 1393

Customizing the privacy policy statement 1394
Customizing the terms of use statement 1394
Editing a site comment 1395
Editing sample content 1395
Editing tags for a specific item 1396
Turning content on or off in the Developer Portal 1396
Deleting a site comment 1396
Deleting blocks 1397
Disabling blogs 1397
Deleting forums from the front page 1398
Turning comments for specific content types on or off in the Developer Portal 1399
Turning comments off for an individual item 1399
Turning off ratings for specific content types in the Developer Portal 1400
Turning the ability to tag specific content types off in the Developer Portal 1400
Structure 1400
Configuring the front page 1401
Disabling blogs 1397
Deleting forums from the front page 1398
Adding a menu 1403
Adding and changing the blocks displayed on Developer Portal pages 1403
Changing the front page banner block 1404
Changing the front page Featured Content block 1405
Changing the items in the main menu 1406
Changing the ratings display 1407
Displaying APIs and Products in categories 1408
Enabling dynamic category creation 1408
Implementing an image carousel 1409
Making your application's image public 1410
Providing navigation by tag hierarchy 1410
Using the Views module in the Developer Portal 1411
Creating views in the Developer Portal 1411
Configuring the search results view in the Developer Portal 1412
Configuring the default number of items in a view list in the Developer Portal 1412
Configuration 1413
Configuring content moderation 1413
Editing, reviewing, and publishing moderated content 1415
General configuration tasks 1415
Adding a page load progress indicator 1416
Adding custom fields to user records 1417
Checking the site status report 1417
Clearing the server caches 1418
Configuring a proxy 1418
Configuring the visibility of analytics in the Developer Portal 1419
Configuring Stripe in the Developer Portal 1419
Configuring cron to run scheduled tasks 1420
Configuring search indexes and servers 1420
Configuring the date and time 1421
Configuring the notifications event log settings 1421
Configuring the settings of a Consumer organization 1422
Configuring the site default timezone 1422
Configuring the site error handling 1422
Configuring which buttons are displayed in the WYSIWYG rich text editor 1423
Configuring which languages are available 1423
Customizing user account settings 1423
Disabling languages 1424
Disabling test tool restrictions 1425
Enabling a cookie compliance banner 1426
Enabling code languages for code snippets 1426
Enabling code snippets for SOAP APIs 1427
Forcing new users to accept terms and conditions 1427
Hiding the admin registry on the login form 1428
Hiding the certificate in the header for APIs secured with mutual TLS 1428
Importing and exporting taxonomies 1429
Restricting access by IP address 1430
Restricting or preventing access from search engines 1432
Toggling the site in and out of maintenance mode 1433
Viewing available updates 1434
Managing Developer Portal security 1434
Configuring CAPTCHA 1435
Configuring reCAPTCHA 1435
Configuring session timeout and limit 1436
Configuring the timeout length for the password reset link 1436

Configuring your site password policy 1437
Disabling CORS warnings 1437
Disabling live testing of APIs 1438
How to enable penetration testing, and information about Developer Portal cookies 1438
How to manage IP security in the Developer Portal 1440
Managing banned IP addresses 1441
Using flood control for login security 1441
Login security 1441
Using the Security kit 1442
Using Honeypot for spam protection 1442
People 1443
Blocking and unblocking specific users 1443
Controlling access to Developer Portal content 1444
Creating a developer account to customize API properties 1444
Email Subscribers 1445
Emailing all subscribers 1445
Emailing api subscribers 1446
Emailing plan subscribers 1447
Emailing product subscribers 1447
Token reference 1448
How to style the Developer Portal emails 1450
Working with roles in the Developer Portal 1451
Creating administrator users for the Developer Portal 1452
Assigning users to a role 1452
Assigning permissions to a role 1453
Creating a new role 1453
Forums 1453
Creating a new forum 1454
Creating new forum containers 1454
Configuring the creation of new forums for each API 1454
Locking forum topics 1455
Marking content in a forum as sticky 1455
Removing forum posts 1455
Turning off forums in the Developer Portal 1456
Reports 1456
Extend 1456
Custom module development: an introduction to Drupal tools for PHP development 1457
Custom module development: background and prerequisites 1458
Custom module development: creating a module skeleton 1459
Custom module development: using hooks 1460
Installing custom modules 1462
Deleting custom modules 1463
Disabling modules 1463
Using Developer Portal metadata on API Manager 1464
Using a custom module for payment method creation 1464
Managing the Developer Portal by using the toolkit CLI 1465
Getting started with the Portal CLI commands 1465
Developer Portal CLI commands 1466
Using the api commands 1470
Using the apic-config commands 1471
Using the application commands 1472
Using the backups commands 1472
Using the consumer-org commands 1473
Using the content commands 1473
Using the custom-module commands 1476
Using the custom-theme commands 1477
Using the custom-translation commands 1477
Using the custom-webserver-page commands 1479
Using the drupal-config commands 1480
Using the drupal-state commands 1482
Using the entity commands 1483
Using the export-entity commands 1483
Using the factory-reset command 1486
Using the forums commands 1486
Using the ip-security-enabled command 1487
Using the modules commands 1488
Using the php-memory commands 1488
Using the platforms commands 1489
Using the product commands 1490
Using the security command 1491
Using the service command 1491
Using the service-ip-allowlist commands 1492

Using the site commands 1492
Using the site-config commands 1495
Using the sites commands 1496
Using the themes commands 1497
Using the twig commands 1498
Scenarios that use the Portal CLI 1498
Exporting and importing custom themes and site configuration 1498
Exporting and importing custom themes and site configuration by using the Portal Admin API 1502
How to enable and disable maintenance mode on your Developer Portal 1509
Troubleshooting guide for the Developer Portal 1510
Migrating your Developer Portal from Version 5 to Version 10 1514
Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10 1516
How to find your custom modules and themes on your Developer Portal 1516
Prepare your Developer Portal for the update for Drupal 10 1520
Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant 1523
Developer Portal tutorials 1526
Tutorial: Creating the Developer Portal 1526
Tutorial: Creating a private Developer Portal site 1530
Tutorial: Creating Accounts and Applications on the Developer Portal 1531
Tutorial: Creating a custom theme for the Developer Portal 1535
Tutorial: Using avatars 1538
Tutorial: Using image optimization 1541
Tutorial: Adding a Back to top button 1544
Tutorial: Creating a drop-down menu link 1545
Tutorial: Displaying blog posts on the front page 1549
Tutorial: Displaying tweets on the front page 1551
Tutorial: Changing the front page of your Developer Portal 1558
Tutorial: Grouping products by category 1564
Tutorial: Adding a field to the sign up form 1567
Tutorial: Adding validation to a field on the sign-up form 1569
Tutorial: Adding a field group to group fields in a content type 1571
Tutorial: Adding a custom block to the front page 1574
Tutorial: Adding a custom block to a page other than the front page 1577
Tutorial: Configuring the RobotsTxt file 1579
Tutorial: Configuring the SecurityTxt file 1580
Tutorials for using custom modules in the Developer Portal 1581
Tutorial: Changing the profile page to display firstname lastname, instead of username 1581
Tutorial: Adding validation to a field on the sign-up form 1569
Tutorial: Using a custom weighting sort order on the product list page 1585
Tutorial: Synchronizing application credentials from the Developer Portal with an external server 1587

Analytics: understand your API usage 1588
Key concepts of API Connect analytics 1588
Preinstallation planning 1590
Estimating storage requirements 1590
Kubernetes and OpenShift preinstallation planning 1591
VMware preinstallation planning 1592
Managing your analytics subsystem 1593
API Connect on VMware 1593
Extra values file reference 1594
API Connect on Kubernetes, OpenShift, and Cloud Pak for Integration 1595
Analytics CR properties reference 1596
OpenSearch nodes, indices, shards, and replicas 1598
Installing the analytics subsystem 1598
Backup, restore, and disaster recovery 1599
Kubernetes, OpenShift, and Cloud Pak for Integration 1599
Taking an analytics infrastructure configuration backup 1600
Configuring analytics database backups 1600
Creating the S3 CA secret 1601
Updating the analytics database backup secret 1602
Running an on-demand analytics database backup 1602
Restoring the analytics database 1603
Identifying specific indices to restore 1605
Recovering from a disaster 1606
Deleting old backups 1606
Updating your S3 object store credentials 1607
Troubleshooting backup and restore 1607
VMware 1608
Backing up your analytics subsystem 1608
Creating the S3 CA secret 1609
Running an on-demand database backup 1610
Restoring an analytics database backup 1611
Recovering from a disaster 1611
API event record field reference 1612
Advanced configuration options 1617
Dedicated storage and scaling up 1617
Horizontal scaling of storage pods 1618

Retention and rollover 1622
Logstash persistent queue 1623
Offload to third-party systems 1624
Configuring offload on Kubernetes and OpenShift 1624
Configuring offload server certificates on Kubernetes and OpenShift 1625
Configuring offload client certificates on Kubernetes and OpenShift 1626
Configuring offload on VMware 1627
Configuring offload server certificates on VMware 1628
Configuring offload client certificates on VMware 1628
Sample output plugins for offloading data 1629
Using offloaded analytics data to train AI models 1631
Troubleshooting offload 1631
Disable local storage 1631
Shutting down the analytics subsystem 1632
Restarting analytics pods 1633
Change deployment profile 1634
Change deployment profile on Kubernetes and OpenShift 1634
Change deployment profile on VMware 1634
Performance tuning with ingestion pipeline properties 1635
Customize your analytics data 1635
Activity log policy 1636
Customizing data with a gateway script or set-variable operation 1637
Customizing data with ingestion filters 1640
Kubernetes and OpenShift individual subsystem installations 1641
Cloud Pak for Integration and OpenShift top-level CR 1642
VMware 1642
Example ingestion filters 1643
Access your analytics data 1644
API Connect UIs 1644
Accessing analytics in the Cloud Manager UI 1644
Accessing analytics in the API Manager UI 1649
Accessing analytics in the Developer Portal 1655
Counting total API calls for your analytics service 1656
Export your analytics data 1656
Allowing API Connect cloud admin users to view provider organization analytics data 1656
Analytics CLI 1657
Analytics REST API 1658
Troubleshooting analytics 1659
No analytics data in UIs 1659
Running out of storage space 1659
Offload failure 1660
Gathering postmortem logs 1660

Reference information 1661
Toolkit command-line tool reference 1661
apic activations 1662
apic activations:clear 1662
apic activations:delete 1662
apic activations:get 1663
apic activations:list 1663
apic analytics 1664
apic analytics:create 1664
apic analytics-services 1665
apic analytics-services:clear 1665
apic analytics-services:create 1666
apic analytics-services:delete 1666
apic analytics-services:get 1667
apic analytics-services:list 1667
apic analytics-services:update 1667
apic api-keys 1668
apic api-keys:create 1668
apic api-keys:delete 1669
apic api-keys:get 1669
apic api-keys:list 1670
apic apis 1670
apic apis:clone 1671
apic apis:document 1671
apic apis:get 1672
apic apis:list 1672
apic apis:list-all 1673
apic apis:update 1673
apic apis:wsdl 1674
apic apps 1674
apic apps:clear 1675
apic apps:create 1675

apic apps:delete 1676
apic apps:get 1676
apic apps:list 1677
apic apps:update 1677
apic associates 1678
apic associates:get 1678
apic associates:list 1679
apic availability-zones 1679
apic availability-zones:clear 1680
apic availability-zones:create 1680
apic availability-zones:delete 1681
apic availability-zones:get 1681
apic availability-zones:list 1681
apic availability-zones:update 1682
apic billings 1682
apic billings:clear 1683
apic billings:delete 1683
apic billings:get 1684
apic billings:list 1684
apic billings:update 1684
apic catalog-settings 1685
apic catalog-settings:get 1685
apic catalog-settings:update 1686
apic catalogs 1686
apic catalogs:clear 1687
apic catalogs:create 1687
apic catalogs:delete 1688
apic catalogs:email-to-owners 1688
apic catalogs:get 1688
apic catalogs:list 1689
apic catalogs:transfer-owner 1689
apic catalogs:update 1690
apic client-creds 1690
apic client-creds:clear 1691
apic client-creds:list 1691
apic client-creds:set 1691
apic cloud-emails 1692
apic cloud-emails:send-to-owners 1692
apic cloud-settings 1692
apic cloud-settings:about 1693
apic cloud-settings:audit-endpoint-test-connection 1693
apic cloud-settings:designer-credentials-list 1694
apic cloud-settings:get 1694
apic cloud-settings:info 1694
apic cloud-settings:mail-server-configured 1695
apic cloud-settings:oauth2-certs 1695
apic cloud-settings:toolkit-credentials-list 1695
apic cloud-settings:topology 1696
apic cloud-settings:update 1696
apic config 1697
apic config:clear 1697
apic config:delete 1698
apic config:get 1698
apic config:list 1699
apic config:set 1700
apic configured-api-user-registries 1701
apic configured-api-user-registries:clear 1701
apic configured-api-user-registries:create 1702
apic configured-api-user-registries:delete 1702
apic configured-api-user-registries:get 1703
apic configured-api-user-registries:list 1703
apic configured-billings 1703
apic configured-billings:clear 1704
apic configured-billings:create 1704
apic configured-billings:delete 1705
apic configured-billings:get 1705
apic configured-billings:list 1706
apic configured-catalog-user-registries 1706
apic configured-catalog-user-registries:create 1707
apic configured-catalog-user-registries:delete 1707
apic configured-catalog-user-registries:get 1708
apic configured-catalog-user-registries:list 1708
apic configured-catalog-user-registries:search 1708

apic configured-gateway-services 1709
apic configured-gateway-services:clear 1709
apic configured-gateway-services:create 1710
apic configured-gateway-services:delete 1710
apic configured-gateway-services:get 1711
apic configured-gateway-services:list 1711
apic configured-oauth-providers 1712
apic configured-oauth-providers:clear 1712
apic configured-oauth-providers:create 1713
apic configured-oauth-providers:delete 1713
apic configured-oauth-providers:get 1713
apic configured-oauth-providers:list 1714
apic configured-tls-client-profiles 1714
apic configured-tls-client-profiles:clear 1715
apic configured-tls-client-profiles:clear-all 1715
apic configured-tls-client-profiles:create 1716
apic configured-tls-client-profiles:delete 1716
apic configured-tls-client-profiles:get 1717
apic configured-tls-client-profiles:list 1717
apic configured-tls-client-profiles:list-all 1718
apic consumer-org-settings 1718
apic consumer-org-settings:delete 1718
apic consumer-org-settings:get 1719
apic consumer-org-settings:update 1719
apic consumer-orgs 1720
apic consumer-orgs:clear 1720
apic consumer-orgs:create 1721
apic consumer-orgs:delete 1721
apic consumer-orgs:get 1722
apic consumer-orgs:list 1722
apic consumer-orgs:transfer-owner 1723
apic consumer-orgs:update 1723
apic create 1724
apic create:api 1724
apic create:product 1725
apic credentials 1726
apic credentials:clear 1726
apic credentials:create 1727
apic credentials:delete 1727
apic credentials:get 1728
apic credentials:list 1728
apic credentials:reset-client-secret 1729
apic credentials:reset 1729
apic credentials:update 1730
apic credentials:verify-client-secret 1730
apic draft-apis 1731
apic draft-apis:clear-all 1731
apic draft-apis:clear 1732
apic draft-apis:clone 1732
apic draft-apis:create 1732
apic draft-apis:delete 1733
apic draft-apis:document 1733
apic draft-apis:get 1734
apic draft-apis:list 1734
apic draft-apis:list-all 1735
apic draft-apis:update 1735
apic draft-apis:update-wsdl 1736
apic draft-apis:validate 1736
apic draft-apis:wsdl 1737
apic draft-products 1737
apic draft-products:clear-all 1737
apic draft-products:clear 1738
apic draft-products:clone 1738
apic draft-products:create 1739
apic draft-products:delete 1739
apic draft-products:document 1740
apic draft-products:get 1740
apic draft-products:list-all 1740
apic draft-products:list 1741
apic draft-products:publish 1741
apic draft-products:update 1742
apic draft-products:validate 1742
apic drafts 1743

apic drafts:clear 1743
apic drafts:list 1744
apic entries 1744
apic entries:clear 1744
apic entries:create 1745
apic entries:delete 1745
apic entries:get 1746
apic entries:list 1746
apic entries:update 1747
apic extensions 1747
apic extensions:clear-all 1748
apic extensions:clear 1748
apic extensions:clone 1749
apic extensions:create 1749
apic extensions:delete 1750
apic extensions:document 1750
apic extensions:get 1751
apic extensions:list 1751
apic extensions:list-all 1752
apic extensions:update 1752
apic gateway-extensions 1753
apic gateway-extensions:create 1753
apic gateway-extensions:delete 1754
apic gateway-extensions:get 1754
apic gateway-extensions:implementation 1755
apic gateway-extensions:update 1755
apic gateway-services 1755
apic gateway-services:clear 1756
apic gateway-services:create 1756
apic gateway-services:delete 1757
apic gateway-services:get 1757
apic gateway-services:list 1758
apic gateway-services:reset-oauth-secret 1758
apic gateway-services:update 1759
apic global-policies 1759
apic global-policies:clear 1760
apic global-policies:clear-all 1760
apic global-policies:create 1761
apic global-policies:delete 1761
apic global-policies:document 1762
apic global-policies:get 1762
apic global-policies:list 1763
apic global-policies:list-all 1763
apic global-policies:update 1764
apic global-policy-errors 1764
apic global-policy-errors:create 1765
apic global-policy-errors:delete 1765
apic global-policy-errors:get 1766
apic global-policy-errors:update 1766
apic global-policy-posthooks 1767
apic global-policy-posthooks:create 1767
apic global-policy-posthooks:delete 1767
apic global-policy-posthooks:get 1768
apic global-policy-posthooks:update 1768
apic global-policy-prehooks 1769
apic global-policy-prehooks:create 1769
apic global-policy-prehooks:delete 1770
apic global-policy-prehooks:get 1770
apic global-policy-prehooks:update 1771
apic groups 1771
apic groups:clear 1772
apic groups:create 1772
apic groups:delete 1773
apic groups:get 1773
apic groups:list 1774
apic groups:update 1774
apic iam-apikey 1775
apic identity-providers 1775
apic identity-providers:list 1775
apic integrations 1776
apic integrations:clear 1776
apic integrations:create 1777
apic integrations:delete 1777

apic integrations:get 1777
apic integrations:list 1778
apic integrations:list-all 1778
apic integrations:update 1778
apic invitations 1779
apic invitations:clear 1779
apic invitations:create 1780
apic invitations:delete 1780
apic invitations:get 1780
apic invitations:list 1781
apic invitations:update 1781
apic jobs 1782
apic jobs:clear 1782
apic jobs:delete 1782
apic jobs:get 1783
apic jobs:list 1783
apic jobs:retry 1784
apic keystores 1784
apic keystores:clear 1784
apic keystores:create 1785
apic keystores:delete 1785
apic keystores:get 1785
apic keystores:list 1786
apic keystores:update 1786
apic licenses 1787
apic log-spec 1787
apic log-spec:get 1787
apic log-spec:update 1788
apic login 1788
apic logout 1789
apic mail-servers 1789
apic mail-servers:clear 1789
apic mail-servers:create 1790
apic mail-servers:delete 1790
apic mail-servers:get 1791
apic mail-servers:list 1791
apic mail-servers:test-connection 1791
apic mail-servers:update 1792
apic me 1792
apic me:change-password 1793
apic me:delete 1793
apic me:get 1793
apic me:update 1794
apic member-invitations 1794
apic member-invitations:clear 1795
apic member-invitations:create 1795
apic member-invitations:delete 1796
apic member-invitations:get 1796
apic member-invitations:list 1797
apic member-invitations:update 1797
apic members 1798
apic members:clear 1798
apic members:create 1799
apic members:delete 1799
apic members:get 1800
apic members:list 1801
apic members:update 1801
apic monitor 1802
apic notification-languages 1802
apic notification-languages:get 1802
apic notification-languages:list 1803
apic notification-languages:update 1804
apic notification-styles 1804
apic notification-styles:get 1805
apic notification-styles:update 1805
apic notification-templates 1806
apic notification-templates:get 1806
apic notification-templates:list 1807
apic notification-templates:list-all 1807
apic notification-templates:update 1808
apic oauth-providers 1809
apic oauth-providers:clear 1809
apic oauth-providers:create 1809

apic oauth-providers:delete 1810
apic oauth-providers:get 1810
apic oauth-providers:list 1811
apic oauth-providers:update 1811
apic org-settings 1812
apic org-settings:get 1812
apic org-settings:update 1812
apic orgs 1813
apic orgs:clear 1813
apic orgs:create 1814
apic orgs:delete 1814
apic orgs:get 1815
apic orgs:list 1815
apic orgs:transfer-owner 1816
apic orgs:update 1816
apic payment-methods 1817
apic payment-methods:create 1817
apic payment-methods:delete 1818
apic payment-methods:get 1818
apic payment-methods:list 1818
apic payment-methods:update 1819
apic permissions 1819
apic permissions:get 1820
apic permissions:list 1820
apic permissions:list-all 1821
apic policies 1821
apic policies:clear 1822
apic policies:clear-all 1822
apic policies:clone 1823
apic policies:create 1823
apic policies:delete 1824
apic policies:document 1824
apic policies:get 1825
apic policies:implementation 1825
apic policies:list 1826
apic policies:list-all 1826
apic policies:update 1827
apic portal-services 1827
apic portal-services:clear 1828
apic portal-services:create 1828
apic portal-services:delete 1829
apic portal-services:get 1829
apic portal-services:list 1830
apic portal-services:update-credentials 1830
apic portal-services:update 1831
apic primary-events 1831
apic primary-events:get 1832
apic primary-events:list 1832
apic products 1833
apic products:clear 1833
apic products:clear-all 1834
apic products:clone 1834
apic products:delete 1835
apic products:document 1835
apic products:execute-migration-target 1836
apic products:get 1836
apic products:list 1837
apic products:list-all 1837
apic products:migrate-subscriptions 1838
apic products:publish 1838
apic products:replace 1839
apic products:set-migration-target 1839
apic products:supersede 1840
apic products:update 1840
apic properties 1841
apic properties:clear 1841
apic properties:create 1842
apic properties:delete 1842
apic properties:get 1843
apic properties:list 1843
apic properties:update 1844
apic registrations 1844
apic registrations:clear 1844

apic registrations:create 1845
apic registrations:delete 1845
apic registrations:get 1846
apic registrations:list 1846
apic registrations:update 1847
apic role-defaults 1847
apic role-defaults:clear 1847
apic role-defaults:create 1848
apic role-defaults:delete 1848
apic role-defaults:get 1849
apic role-defaults:list 1849
apic role-defaults:list-all 1850
apic role-defaults:update 1850
apic roles 1851
apic roles:clear 1851
apic roles:create 1852
apic roles:delete 1852
apic roles:get 1852
apic roles:list 1853
apic roles:update 1854
apic services 1854
apic services:clear 1854
apic services:clear-all 1855
apic services:create 1855
apic services:delete 1856
apic services:get 1856
apic services:list 1857
apic services:list-all 1857
apic services:update 1858
apic space-settings 1858
apic space-settings:get 1859
apic space-settings:update 1859
apic spaces 1860
apic spaces:clear 1860
apic spaces:create 1860
apic spaces:delete 1861
apic spaces:email-to-owners 1861
apic spaces:get 1862
apic spaces:list 1862
apic spaces:transfer-owner 1863
apic spaces:update 1863
apic subscriber-events 1864
apic subscriber-events:get 1864
apic subscriber-events:list 1865
apic subscriptions 1865
apic subscriptions:clear 1866
apic subscriptions:create 1866
apic subscriptions:delete 1867
apic subscriptions:get 1867
apic subscriptions:list 1868
apic subscriptions:update 1868
apic task-queues 1869
apic task-queues:get 1869
apic task-queues:list 1870
apic tasks 1870
apic tasks:get 1871
apic tasks:list 1871
apic tasks:update 1872
apic test 1872
apic tls-client-profiles 1872
apic tls-client-profiles:clear 1873
apic tls-client-profiles:clear-all 1873
apic tls-client-profiles:create 1874
apic tls-client-profiles:delete 1874
apic tls-client-profiles:get 1875
apic tls-client-profiles:list 1875
apic tls-client-profiles:list-all 1875
apic tls-client-profiles:update 1876
apic tls-server-profiles 1877
apic tls-server-profiles:clear 1877
apic tls-server-profiles:clear-all 1877
apic tls-server-profiles:create 1878
apic tls-server-profiles:delete 1878

apic tls-server-profiles:get 1879
apic tls-server-profiles:list 1879
apic tls-server-profiles:list-all 1880
apic tls-server-profiles:update 1880
apic truststores 1880
apic truststores:clear 1881
apic truststores:create 1881
apic truststores:delete 1882
apic truststores:get 1882
apic truststores:list 1883
apic truststores:update 1883
apic user-registries 1884
apic user-registries:clear 1884
apic user-registries:create 1884
apic user-registries:delete 1885
apic user-registries:execute 1885
apic user-registries:get 1886
apic user-registries:list 1886
apic user-registries:search 1887
apic user-registries:test-connection 1887
apic user-registries:update 1887
apic user-registry-settings 1888
apic user-registry-settings:get 1888
apic user-registry-settings:update 1889
apic users 1889
apic users:clear 1890
apic users:create 1890
apic users:delete 1891
apic users:get 1891
apic users:list 1891
apic users:request-password-reset 1892
apic users:search-provider 1892
apic users:update 1893
apic validate 1893
apic version 1894
apic webhooks 1894
apic webhooks:get 1895
apic webhooks:list 1895
apic webhooks:update 1896
apic wsdl 1896
apic wsdl:introspect 1896
Portal command-line tool reference 1897
apic api 1897
apic api:add-attachment 1897
apic api:add-tag 1898
apic api:get 1898
apic api:get-document 1898
apic api:list 1899
apic api:set-icon 1899
apic apic-config 1900
apic apic-config:get 1900
apic application 1900
apic application:get 1901
apic application:list 1901
apic backups 1901
apic backups:list 1902
apic config 1902
apic config:clear 1902
apic config:delete 1903
apic config:get 1904
apic config:list 1904
apic config:set 1905
apic consumer-org 1906
apic consumer-org:get 1906
apic consumer-org:list 1906
apic content 1907
apic content:create-export 1907
apic content:create-import 1908
apic content:delete-export 1908
apic content:delete-import 1908
apic content:get-export-status 1909
apic content:get-export 1909
apic content:get-import-status 1909

apic content:list 1910
apic content:list-types 1910
apic custom-module 1911
apic custom-module:create-export 1911
apic custom-module:create-import 1911
apic custom-module:delete-export 1912
apic custom-module:delete-import 1912
apic custom-module:get-export 1912
apic custom-module:get-export-status 1913
apic custom-module:get-import-status 1913
apic custom-theme 1914
apic custom-theme:create-export 1914
apic custom-theme:create-import 1914
apic custom-theme:delete-export 1915
apic custom-theme:delete-import 1915
apic custom-theme:get-export 1915
apic custom-theme:get-export-status 1916
apic custom-theme:get-import-status 1916
apic custom-translation 1916
apic custom-translation:create-export 1917
apic custom-translation:create-import 1917
apic custom-translation:delete-export 1918
apic custom-translation:delete-import 1918
apic custom-translation:get-export 1918
apic custom-translation:get-export-status 1919
apic custom-translation:get-import-status 1919
apic custom-webserver-page 1919
apic custom-webserver-page:delete 1920
apic custom-webserver-page:get 1920
apic custom-webserver-page:set 1920
apic drupal-config 1921
apic drupal-config:delete 1921
apic drupal-config:get 1922
apic drupal-config:list 1922
apic drupal-config:set 1922
apic drupal-state 1923
apic drupal-state:delete 1923
apic drupal-state:get 1923
apic drupal-state:set 1924
apic drupal-state 1923
apic drupal-state:delete 1923
apic drupal-state:get 1923
apic drupal-state:set 1924
apic entity 1926
apic entity:count 1926
apic export-entity 1926
apic export-entity:add-content 1927
apic export-entity:create 1927
apic export-entity:delete-launch-export 1927
apic export-entity:delete 1928
apic export-entity:get-launch-export-status 1928
apic export-entity:get-launch-export 1928
apic export-entity:get 1929
apic export-entity:launch 1929
apic export-entity:list 1930
apic export-entity:remove-content 1930
apic factory-reset 1930
apic factory-reset:delete 1931
apic forums 1931
apic forums:disable 1931
apic forums:enable 1932
apic ip-security-enabled 1932
apic ip-security-enabled:update 1932
apic licenses 1933
apic modules 1933
apic modules:delete 1934
apic modules:disable 1934
apic modules:enable 1934
apic modules:list 1935
apic php-memory 1935
apic php-memory:list 1935
apic php-memory:update 1936
apic platforms 1936

apic platforms:list 1937
apic product 1937
apic product:add-attachment 1937
apic product:add-tag 1938
apic product:get 1938
apic product:get-document 1938
apic product:list 1939
apic product:set-icon 1939
apic security 1940
apic security:clear-bans 1940
apic service 1940
apic service:list 1941
apic service-ip-allowlist 1941
apic service-ip-allowlist:add 1941
apic service-ip-allowlist:delete 1942
apic service-ip-allowlist:list 1942
apic service-ip-allowlist:remove 1942
apic site 1943
apic site:cache-rebuild 1943
apic site:check 1943
apic site:create-export 1944
apic site:create-import 1944
apic site:delete-export 1944
apic site:delete-import 1945
apic site:get-export-status 1945
apic site:get-export 1946
apic site:get-import-status 1946
apic site:login-link 1946
apic site:state 1947
apic site-config 1947
apic site-config:create-export 1947
apic site-config:create-import 1948
apic site-config:delete-export 1948
apic site-config:delete-import 1948
apic site-config:get-export 1949
apic site-config:get-export-status 1949
apic site-config:get-import-status 1950
apic sites 1950
apic sites:check 1950
apic sites:list 1951
apic themes 1951
apic themes:delete 1951
apic themes:disable 1952
apic themes:enable 1952
apic themes:list 1953
apic themes:set-default 1953
apic twig 1953
apic twig:debug-disable 1954
apic twig:debug-enable 1954
apic twig:debug-status 1954
apic version 1955
API Connect REST APIs 1955
Example: Using the platform REST APIs to publish a product containing a SOAP API 1955
Example Product file: globalweatherproduct_1.0.0.yaml 1957
Example API definition: globalweather_1.0.0.yaml 1957
Example WSDL file: globalweather.wsdl 1962
API Connect TLS certificates 1965
TLS certificates organized by usage 1968
Troubleshooting API Connect 1973
Unable to access Management UIs 1973
Unable to access the portal sites 1974
Login failure on Management UIs 1975
Login failure on Portal sites 1975
Management operations not reflected on Gateway or Portal 1976
Externally linked support topics 1977
Management subsystem Postgres failover steps V10.0.6.0 1977
Management database changes V10.0.7.0 1979
Restore blocked on 2DCDR deployments 1979

API Connect overview

IBM® API Connect is an integrated API management offering, with capabilities and tooling for all phases of the API lifecycle. IBM API Connect 10.0.7.0 is the latest 10.0.x Continuous Delivery (CD) release, following-on from the previous 10.0.6.0 CD release. If you're looking for a different version of API Connect, use the Change version list in the navigation pane to select the version that you require.

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

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

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

Table 1. Key phases of the API lifecycle

Lifecycle Phase	Description
Create	Develop and write API definitions from an API development environment, eventually bundling these APIs into consumable products, and deploying them to production environments. For tutorials, walk-throughs, and in-depth guides for building, testing, and deploying APIs and Products in API Connect, see Tutorials, and Developing your APIs and applications.
Secure	Leverage the best-in-class API Gateway, gateway policies, and more, to manage access to your APIs and back-end systems. To learn more about adding security to your API, see *Enforcing security requirements on an API* for the OpenAPI version that you are working with: • OpenAPI 2.0: Enforcing security requirements on an API • OpenAPI 3.0: Enforcing security requirements on an API To learn more about how to add API Gateway policies to your API, see API policies and logic constructs.
Manage	Governance structures are built in to the entire API lifecycle, from managing the view/edit permissions of APIs and Products being deployed, to managing what application developers can view and subscribe to when APIs are deployed. To understand and leverage API Connect management and governance controls along the API lifecycle, see Managing your APIs.
Socialize	API Connect comes with an advanced Developer Portal that streamlines the onboarding process of application developers, and can be completely customized to an organization's marketing standards. To understand more about using the Developer Portal, see Developer Portal: Socialize your APIs.
Monetize	Bundle APIs into Products and Plans with monetization schemes that are built into those Plans, thereby opening up digital revenue streams. To understand more about how to leverage monetization strategies in API Connect, see Monetizing your Products.
Analyze	Developers and Product Managers alike are given the tooling in API Connect to understand their API traffic patterns, latency, consumption, and more to make data driven insights into their API initiatives. To learn more about how to leverage API Connect analytics tooling, see API Analytics.

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

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

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

- Packaging strategy and terminology in API Connect
- API Connect components

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

- **What's new in the latest release (Version 10.0.7.0)**
  Find out about the newest features and the latest updates in API Connect.
- **Language versions offered by API Connect**
  API Connect is offered in several languages, as enabled by the operating platform on which the product is installed.
- **Known limitations**
  This page describes the known limitations for API Connect 10.0.7.0 and later.
- **API Connect concepts**
  To help you get started, read about the API Connect concepts and obtain a high level understanding of the API management solution.
- **API Connect glossary**
  The IBM API Connect and Cloud Manager glossary of terms and definitions.
- **Accessibility features for IBM API Connect**
  Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.
- **Legal information**
  Notices, and terms and conditions for information centers.
- **IBM API Connect Considerations for GDPR Readiness**
  Information about features of IBM API Connect that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness.
- **Essential reading**
  These articles by IBM API Connect product specialists provide a wealth of supporting information on APIs and the API economy.

What's new in the latest release (Version 10.0.7.0)

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

IBM® API Connect 10.0.7.0 is the latest 10.0.x Continuous Delivery (CD) release, following-on from the previous 10.0.6.0 CD release. IBM API Connect 10.0.7.0 CD includes the following enhancements.

Product files and release notes

- Access the latest files from IBM Fix Central by searching for the API Connect product and your installed version. Full installation files for IBM API Connect can be downloaded from Passport Advantage.
- For details on the specific APARs that are included in this release, links to downloads, and additional blogs and conference notices, see the IBM API Connect 10.0.7.0 Support Announcement page.

What's new for Developers

LoopBack is deprecated
: LoopBack is now deprecated and support will be removed in an upcoming release.

New toolkit command to execute API tests stored in a YAML file
: The API Connect developer toolkit includes a new command **`apic test`** that uses the automated API behavior testing application to execute API tests. You can include one or more YAML files in the command, and each file can contain one or more API tests to be executed. For more information, see Using the toolkit CLI to execute API tests.

LDAP updates

- The UI for creating an LDAP registry now provides options for specifying the scope for "Search DN" (subtree, one level, and base) as you can in the CLI.
- You can now specify whether your LDAP is Microsoft Active Directory in the UI and with the CLI, to ensure that the directory is handled correctly in API Connect.

For more information, see Creating an LDAP user registry andUsing the CLI to create an LDAP user registry.

Updates to the assembly authoring experience in the API editor
: The API editor for viewing and editing the assembly section of APIs in the API Manager and API Designer is updated with a new user experience. The overall capability remains the same, but the visual components for working with the flow canvas, the policy palette, and the policy properties are updated. For more information, see Including elements in your API assembly.

New rate limit and GraphQL built-in policies
: The following rate limit and GraphQL policies are added to the API assembly editor:

- **`ratelimitinfo`**
- **`graphql-execute`**

Improvements to the API testing experience in the API editor
: The Test tab section of the API editor is updated to allow API testing independently of the auto-publish setting, and to make important selections more visible. For more information, see Using the Test tab to debug your API.

What's new for API product managers

New timeout setting available for the self service onboarding task
: You can now configure the timeout period for the self service onboarding task of API consumers into a catalog, and the associated Developer Portal. Previously, the self service onboarding task timeout was set at 72 hours, and couldn't be changed. Now you can update this setting by using the Self service onboarding task timeout setting in the Onboarding section of the Catalog settings tab. Note that this timeout setting includes the activation link and, if Self service onboarding approval is selected, the approval process as well. For more information, see Creating and configuring catalogs.

API governance service updates
: The following updates to the API governance service are now available:

- Validating API documents is now available in API Designer when the UI is online, and is connected to a cloud instance that has the API governance microservice enabled on it.
- When creating and editing rulesets, you can now add your own version number information, and can have rulesets with the same name but with different version numbers. Note that version numbers must be of the format **`major.minor.patch`**, for example **`1.0.0`**.
- The ruleset Name field is now auto-generated based on the Title field.
- The version of the Spectral rulesets now matches the version of that ruleset that's available in Spectral. Spectral ruleset names are prefixed by spectral-.
- You can now use the toolkit CLI to configure API governance in the Cloud Manager and the API Manager. You can also use the commands to validate an API document. For a complete list of the commands, and information about how to run them, run the following commands in the toolkit CLI:
  - **`apic --mode governance rulesets --help`** - displays the commands that are available for creating and managing rulesets.
  - **`apic --mode governance rules --help`** - displays the commands that are available for creating and managing rules.
  - **`apic --mode governance compliance --help`** - displays the commands that are available for running validation on API documents.

For more information about the API governance service, see Configuring API governance in the API Manager and Validating an API document by using API governance.

Provider organization analytics data is not shared with cloud admin users
: In previous releases of API Connect, Provider organization-specific analytics data was available to Cloud Admins by default. Beginning with version 10.0.7.0, the data is not shared with Cloud Admins by default; the owner of a Provider organization can optionally enable the sharing of analytics data with Cloud Admins. The new setting does not apply to total API call volume data; Cloud Admins can still see total API calls across the API Connect deployment. For more information, see Allowing cloud admins to view provider organization analytics data.

Analytics Monitoring Data Dashboard
: The Cloud Manager and API Manager analytics view have a new dashboard called Monitoring Data Dashboard. The new dashboard provides information on which applications, plans, consumer organizations, and APIs are sending and receiving the most data.

New analytics API event fields
: When using API Gateway v10.5.3 or higher, two new fields are included in API event records:

- **`api_resource_id`**: String containing the resource ID for the API used by the gateway. Format is **`api_name:api_version:method:path`**.
- **`gateway_service_name`**: The name of the gateway service, as configured in the Cloud Manager UI.

Analytics event **`query_string`** fields are now stored as text
: API event **`query_string`** fields are now stored in analytics as text for improved indexing.

Top 20 analytics charts
: New analytics charts that show the top 20 APIs, applications, and consumer organizations.

Analytics scroll API responses sorted by datetime
: Calls to the analytics REST API **`events/scroll`** operation return results sorted by datetime.

Analytics inactive product report
: New report in the API Manager UI that highlights products that have no subscriptions, no recent traffic, or no traffic at all.

Analytics inactive consumer report
: New report in the API Manager UI that highlights consumers that have no applications, no subscriptions, no recent traffic, or no traffic at all.

Various analytics UI enhancements

- UI Dashboards and Discover view has auto-refresh option to refresh the UI every 30 seconds automatically.
- The columns displayed in the Analytics UI Discover view are now configurable and can be saved as part of a saved/shared query.
- Calendar widget for easier date selection.
- Display total option on specific charts. For example, when the option is enabled, the Top APIs chart shows the total API calls, so the comparison of calls to a single API against the total API calls is clear.
- Shared and saved query table includes column showing filters used, for easier identification of the desired query.
- New time range options for viewing API event data in the UI:
  - Last minute of API event data.
  - Last 5 minutes of API event data.
- The time axis on analytics charts is sized appropriate to the available data points and selected time period.

Analytics API call volume leaderboards
: The reports tab in the analytics view includes a leaderboard of the top APIs, products, plans, applications, and consumer organizations.

Analytics consumer trend report
: The reports tab in the analytics view includes a new consumer trend report that shows changes to the number of applications and subscriptions in consumer organizations over time.

Detailed API, Product, Plan, Application, and consumer organization analytics reports
: Detailed analytics information about specific APIs, products, plans, applications, and consumer organizations. For example, for an API, details of all the consumer organizations that use it. The detailed reports are accessible from the leaderboards in the reports view.

What's new for Developer Portal site administrators

New Developer Portal **service** command
: The **service** command enables you to list the Developer Portal service that is currently installed. For more information, see Using the service command.

New Developer Portal **content** commands
: The new **content** commands allow you to list, export and import your Developer Portal site content. The following **content** commands are now available:

- **`content:create-export`**
  Creates a task to export a .tgz file of your site content.
- **`content:create-import`**
  Creates a task to import an archive of your site content.
- **`content:delete-export`**
  Cancels any currently running **`content:create-export`** tasks, and deletes any related artifacts.
- **`content:delete-import`**
  Cancels any currently running **`content:create-import`** tasks, and deletes any related artifacts.
- **`content:get-export`**
  Streams the content of a specific completed export task to a .tgz file.
- **`content:get-export-status`**
  Returns the status of a specific export task.
- **`content:get-import-status`**
  Returns the status of a specific import task.
- **`content:list-types`**
  Lists the exportable content types on your site.
- **`content:list`**
  Lists all of the entities on your Developer Portal site for the given content type and bundle.

For more information about the **content** commands and how to use them, see Using the content commands.

New Developer Portal **export-entity** commands
: The following **export-entity** commands are added, which enable you to export assorted entity content from your Developer Portal site.

- **`export-entity:create`**
  Creates a new export entity, which is the container for the entity content that you want to export.
- **`export-entity:add-content`**
  Adds content to an existing export entity.
- **`export-entity:get`**
  Returns a list of the content of a specific export entity.
- **`export-entity:remove-content`**
  Removes certain content from a specific export entity.
- **`export-entity:delete`**
  Deletes a specific export entity.
- **`export-entity:launch`**
  Launches an export entity polling task that creates a .tgz file of all of the entities that are contained in a specific export entity. Can be run with a **`--no-poll`** option, in which case the task doesn't return a .tgz file, but just returns the task ID.
- **`export-entity:get-launch-export`**
  Streams the content of a specific completed **`export-entity:launch`** task to a .tgz file.
- **`export-entity:delete-launch-export`**
  Cancels a currently running **`export-entity:launch`** task, and deletes any related artifacts.
- **`export-entity:get-launch-export-status`**
  Returns the status of a specific **`export-entity:launch`** task.
- **`export-entity:list`**
  Returns a list of all of the export entities within a specific Developer Portal. Each export entity contains a defined list of all of the entity content that will be exported if **`export-entity:launch`** is run.

For more information about the **export-entity** commands and how to use them, see Using the **export-entity** commands.

Note that you can now also export and import entities from the Developer Portal UI. When you're editing a content entity type in the UI, you can click Export in the side navigation bar. To create an export entity container and export that entity, including any required embedded entities, click Export entity. Or, if you have an existing export entity container, you can select the required container, and click Add to the export. You can also view and manage all of your export entities by clicking Content > Content synchronizer > Exports, and all of your import entities by clicking Content > Content synchronizer > Imports.

Updates to the Developer Portal **site** commands

The **site** command now enables you to export and import the entire configuration for a Developer Portal site, including custom modules, custom themes, site configuration, and site content. The added commands mean that you can easily replicate a Developer Portal site, for example replicating a test site into a production site. The following **site** commands are added:

- **`site:create-export`**
  Creates a task to export a .tgz archive file of your entire site configuration. You can then use this archive to create an identical Developer Portal site.
- **`site:create-import`**
  Creates a task to import an archive of your entire site configuration, completely overriding the original site configuration, including content, custom modules, and custom themes.
- **`site:delete-export`**
  Cancels any currently running **`site:create-export`** tasks, and deletes any related artifacts.
- **`site:delete-import`**
  Cancels any currently running **`site:create-import`** tasks, and deletes any related artifacts.
- **`site:get-export-status`**
  Returns the status of a specific export task.
- **`site:get-export`**
  Streams the content of a specific completed export task to a .tgz file.
- **`site:get-import-status`**
  Returns the status of a specific import task.

Important: If you want to import a site export configuration file, the export file must have been created on the same version of API Connect as the version that you want to import to.
For more information about the **site** commands and how to use them, see Using the **site** commands.

Updates to the list of blocked Drupal modules

The following Drupal modules are now unsupported and their installation is blocked in the Developer Portal:

- All of the advanced aggregation modules; **`advagg`**, **`advagg_mod`**, **`advagg_js_minify`**, **`advagg_css_minify`**, **`advagg_ext_minify`**, **`advagg_validator`**, and **`advagg_bundler`**. These modules are blocked due to incompatibility with the current Drupal version.
- **`statistics`** module. This module is being deprecated by Drupal.
- **`tfa`** module. Two-factor authentication isn't available within the Developer Portal. If multi-factor authentication is required, it can be configured within an OpenID Connect (OIDC) user registry; see Creating an OIDC user registry.

For more information about custom modules, see Installing custom modules.

Ability to identify the realm parameter when logging in as a consumer to the toolkit CLI

You can now find out which realm parameter you need to use when logging in to the Developer Portal with the toolkit CLI, by running the following command:

```
apic identity-providers:list --server consumer_endpoint_api --mode consumer --catalog catalog_name_or_id --org
<provider_org_name_or_id> --fields registry_type,realm
```

For more information, see Logging in as a consumer to the Developer Portal by using the CLI.

Ability to configure the analytics chart views in the Developer Portal

You can now configure which analytics charts of API data are displayed to API consumers in the Developer Portal. Previously, if access to analytics data is granted, API consumers see all of the default charts of application and organization analytics data, including API statistics, response times, and error information. Now, you can configure which charts are displayed to your API consumers, by using the Configuration > System > IBM API Developer Portal Consumer Analytics menu in the Developer Portal UI.

For more information, see Configuring analytics in the developer portal.

What's new for DevOps

Local and SFTP management database backup not available in v10.0.7.0

In v10.0.7.0, the management database can be backed up to an S3 object-store only. Support for SFTP and local backups will be provided in a future v10.0.x release.

If you are on v10.0.6.0 and using SFTP or local backups for your management database, then to upgrade to v10.0.7.0 you have two options:

- Update your management database backup configuration to use an S3 object-store before you start the upgrade process.
- Enable an opt-out setting to allow upgrade to v10.0.7.0 without management database backups configured.

For more information, see the upgrade steps for your platform:

- Upgrading on Kubernetes, OpenShift, and Cloud Pak for Integration
- Upgrading on VMware

New analytics deployment profile

A new analytics profile is available on all platforms: n3xc4.m32. Use this profile instead of the existing n3xc4.m16 profile if you have a high analytics load, since the 16 Mi of memory in the `n3c4.m16` profile can be insufficient.

Updates to analytics deployment profile storage and ingestion pods

The memory requests and limits of the storage pods is reduced for the following profiles:

- `n1xc6.m48`: Reduced from 38 Gi to 37 Gi.
- `n3xc6.m48`: Reduced from 38 Gi to 37 Gi for shared storage, and from 36 Gi to 35 Gi for dedicated storage.
- `n3xc8.m64`: Reduced from 54 Gi to 53 Gi for shared storage, and from 50 Gi to 49 Gi for dedicated storage.

The default PVC size for the ingestion pod is increased from 5 Gi to 50 Gi in all profiles.

Analytics ingestion resiliency

Analytics persistent queue feature updated for better resiliency when both internal storage and offload is configured. The offload processes are now separated from the internal storage processes.

Replacement of management database operator

New Postgres operator on the management component: EDB.

EDB replaces the previous Postgres operator Crunchy.

The change to EDB results in some changes to the procedures for:

- Install.
- Upgrade.
- Backup, restore, and disaster recovery.
- Form factor migration.
- Two data center disaster recovery.
- Management database maintenance and monitoring.

Analytics persistent queue is enabled by default

The persistent queue feature is enabled by default on new v10.0.7.0 installations. If you are upgrading from v10.0.5.x or v10.0.6.0, the feature is automatically enabled during upgrade.

Cloud Pak endpoints are deprecated for API Connect

Beginning with version 10.0.7.0, API Connect no longer uses the Cloud Pak `cpd` routes for endpoints when deployed as a component of Cloud Pak for Integration. Instead, the component uses the typical default API Connect routes (or the custom endpoints configured in the CR). This change affects both new installations and upgrades from previous versions of the API Connect component in Cloud Pak for Integration.

If you want to deploy the API Connect component with Cloud Pak endpoints, or you need to preserve your existing endpoints (for example, to support existing bookmarks and automation features), you can enable the use of Cloud Pak endpoints when installing or upgrading the API Connect component in Cloud Pak for Integration 2023.4.1 or later. For more information, see Deploying on OpenShift and Cloud Pak for Integration or Upgrading on OpenShift and Cloud Pak for Integration.

Technical Preview: New API Connect Config Sync utility to replicate consumer-side catalog data

API Connect Config Sync is a utility that can be run either as a standalone binary, or as part of a Kubernetes cronjob to facilitate the unidirectional replication of consumer-side data (consumer organizations, members, apps, subscriptions, credentials) from a catalog in a source API Connect cluster to a corresponding catalog in a separate, target API Connect cluster.

For more information, see Using API Connect Config Sync to replicate consumer-side catalog data.

VMware: New apicup command to get node status from your local machine

The new `apicup subsys status` command can be run locally to get a node's status without requiring you to first SSH into the node.

Cert-manager upgraded to version 1.11.5

API Connect 10.0.7.0 uses cert-manager 1.11.5. If your environment requires a manual installation or upgrade of cert-manager, the instructions are included as part of the API Connect installation and upgrade procedures.

Update to enabling API governance on VMware

Previously, you had to enable the API governance microservice by updating an extra-values file. Now you can enable the microservice by running the following command:

`apicup subsys set mgmt_subsystem_name governance-enabled=true`

Where ***mgmt_subsystem_name*** is the name of the management subsystem that you are configuring.
The governance microservice is set to **false** by default.

Updates to the Developer Portal local backup process
: The Developer Portal now displays local backups, as well as remote backups, when the following command is run:

`kubectl get portalbackup`

Previously, local backups were visible only inside the portal pod.
Note that the Developer Portal retains only three system backups, and three backups per site, for local backups. Running a new local backup will cause the oldest backup to be deleted.

Enhancements to the Developer Portal caching process
: The Developer Portal now has enhanced in-memory caching, which increases the speed of the page accesses for the Developer Portal web sites. This improved site performance is particularly helpful for long running administrative tasks, such as enabling and disabling modules. However, it is possible to disable these enhancements if required; see Enabling Developer Portal feature flags on Kubernetes, Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration, or Enabling Developer Portal feature flags on VMware for information.

New backup, restore, and disaster recovery documentation
: For Kubernetes, OpenShift, and Cloud Pak for Integration, the disaster recovery section is merged with the backup and restore section, and the requirement to backup the subsystem YAML files and Kubernetes secrets is more prominent. See Backing up, restoring, and disaster recovery.
For VMware, there is a new management subsystem backup section for v10.0.7.0. See Backing up and restoring the management subsystem.

Upgrade improvements
: Updates to the analytics microservices are redesigned, leading to reduced downtime during upgrades.

What's new for security practitioners

Support for new OIDC protocol in LinkedIn
: If you create a user registry for API Connect using LinkedIn as your OIDC provider, note that LinkedIn updated their OIDC protocol. The changes affect how you configure the OIDC registry in API Connect, and are explained in Creating an OIDC user registry in API Manager and Configuring an OIDC user registry in Cloud Manager.

LDAP updates
: - The UI for creating an LDAP registry now provides options for specifying the scope for "Search DN" (subtree, one level, and base) as you can in the CLI.
- You can now specify whether your LDAP is Microsoft Active Directory in the UI and with the CLI, to ensure that the directory is handled correctly in API Connect.

For more information, see Configuring an LDAP user registry in the Cloud Manager and Using the CLI to configure a shared LDAP user registry .

API key now supports multiple uses
: When defining the API key timeout in Cloud Manager, you can additionally choose whether to allow an application to exchange the API key for an access token multiple times. For more information, see Configuring API key settings.

OpenShift: Support for FIPS configuration on the API Connect cluster
: For new deployments, you can configure support for the Federal Information Processing Standards (FIPS) protocol on the cluster. You must configure FIPS support before installing the OpenShift cluster. For more information, see Configuring FIPS support.

Cloud Pak for Integration replaces IAM with Keycloak as the OIDC provider and user accounts might require updates
: Starting with 2023.4 (API Connect 10.0.7.0), Cloud Pak for Integration uses Keycloak as an OIDC provider to authenticate users instead of IAM (Identity and Access Management). Due to differences in how Keycloak and IAM treat user names, you might need to manually merge duplicate user accounts to ensure users can log in after the upgrade. For more information, see Resolving duplicate users before upgrading on Cloud Pak for Integration.

- **What's new in Version 10.0.6.0**
  This page details the enhancements in IBM API Connect Version 10.0.6.0 since the previous release.

What's new in Version 10.0.6.0

This page details the enhancements in IBM® API Connect Version 10.0.6.0 since the previous release.

What's new for Developers

New API governance add-on service
: API governance is an optional add-on to IBM API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process. For more information, see Validating an API document by using API governance.

Updated schema editor design for API definitions
: The Definitions section of the API editor in the API Manager and API Designer UIs is updated to improve the user experience when creating and editing schema definitions. For more information, see Defining schemas for an API.

What's new for API product managers

New API governance add-on service
: API governance is an optional add-on to IBM API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process. For more information, see Configuring API governance in the Cloud Manager, and Configuring API governance in the API Manager.

Analytics long-term data

By default analytics event data is retained for up to 30 or 90 days, dependent on the deployment profile. With the new long-term data feature, event data is summarized and added to a new summary data index that has a 1 year default retention period. The summary data allows the product manager to view long-term trends. To view the summary data, the UIs have a new Reports tab, see Analytics dashboards. New REST API and toolkit CLI operations are also provided to access the summary data shown in the UI Reports tab.
Note: If you are upgrading from v10.0.5, the summary data is gathered for all new API events after upgrade to 10.0.6 is complete. Existing API event data from 10.0.5 is not added to the summary data index.

Gateway policy titles, names, and types included in assembly latency visualization

DataPower® v10.5.1.x only: To help identify where delays in API processing occur, when you click an API event in the analytics UI Discover view, the title, name, and type of gateway policies are included.

What's new for Developer Portal site administrators

New Developer Portal CLI product commands

The new **product** commands allow you to update the images, attachments, tags and categories for products on your Developer Portal. In IBM API Connect 10.0.6.0 the newly added **product** commands are:

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

For more information about the **product** commands and how to use them, see Using the product commands.

New Developer Portal CLI api commands

The new **api** commands allow you to update the images, attachments, tags, and categories for products on your Developer Portal. In IBM API Connect 10.0.6.0 the newly added **api** commands are:

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

For more information about the **api** commands and how to use them, see Using the api commands.

New Developer Portal CLI site command

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

For more information about the **site** commands and how to use them, see Using the site commands.

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

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

For more information about the `--custom` flag, see Using the themes commands and Using the modules commands.

What's new for DevOps

Migrate v5-compatible APIs to the DataPower API Gateway

If your deployment includes v5-compatible APIs, they can only be used with the DataPower v5-compatible Gateway (also known as the v5c gateway). To use the v5-compatible APIs with the DataPower API Gateway, you must migrate the APIs to the newer format. For more information, see Migrating v5-compatible APIs to API Gateway.

Direct upgrades are only supported from API Connect 10.0.5.x

API Connect 10.0.6.0 supports a direct upgrade only from versions of 10.0.5.x. If your deployment is using an older release of API Connect, you must upgrade to a 10.0.5.x release before attempting to upgrade to 10.0.6.0. For more information about upgrading, see:

- Upgrading on Kubernetes
- Upgrading on OpenShift and Cloud Pak for Integration
- Upgrading on VMware

Cert-manager upgraded to version 1.9.1

API Connect 10.0.6.0 uses cert-manager 1.9.1. If your environment requires a manual installation or upgrade of cert-manager, the instructions are included as part of the API Connect installation and upgrade procedures.

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

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

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

In previous releases of API Connect, a deployment on Red Hat OpenShift (including as a capability in Cloud Pak for Integration) managed the use of IBM Cloud Pak foundational services through dependencies that are enforced by OLM at the operator level. In API Connect 10.0.6.0, the dependency on foundational services is no longer managed at the operator level by OLM, but is instead handled at the operand level by the API Connect operator. This change provides more flexibility going forward by allowing the APIC operator to enforce the dependency only on the operands that actually require it.
This change applies only to new installations and upgrades from 10.0.5.3 (or later). For more information, see the appropriate installation instructions for your Red Hat OpenShift environment.

Upgrade checks for Postgres leader

The upgrade process on all platforms now includes checks for the condition where the original Postgres leader pod is no longer the leader, or if there is no Postgres leader pod. In both cases extra steps are required to complete the management subsystem upgrade: Postgres failover steps.

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

What's new for security practitioners

New API governance optional add-on service
: API governance is an optional add-on to IBM API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process. For information about how to enable this add-on, see Enabling API governance on Kubernetes, and Enabling API governance on VMware.

Language versions offered by API Connect

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

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

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

Known limitations

This page describes the known limitations for API Connect 10.0.7.0 and later.

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

Gateway services that span data centers
: If you want to configure a gateway service that consists of gateways in different data centers, then note the following points:

  - This configuration should be used only if TMS is required and an external OAuth provider cannot be used.
  - Network latency between sites must be less than 30 ms.
  - The analytics service cannot span data centers. If you want to associate an analytics service with your gateway service, you must choose one of the data centers to host the analytics service.

Cloud Pak for Integration: Cannot add an API to the Asset Repository
: If you are using API Connect as a component of Cloud Pak for Integration, you cannot add an API to the Asset Repository from API Connect.

Cloud Pak for Integration: Problem importing from Asset repository when user remains inactive
: If there is no user activity on the page while importing an API from the Cloud Pak for Integration Asset repository, the user might see the following error: **`could not add asset, parent window missing`**
Work around the issue by reloading the Import API page and then opening the Asset repository again.

Possible issues with AutoTest
: If you observe the AutoTest profile not creating, or the management-turnstile pod crashing, you can work around the issue by applying the override to the Management CR:

```
    - name: turnstile
      containers:
        - name: turnstile
          resources:
        limits:
          memory: 96Mi
        requests:
          memory: 48Mi
```

Limitations to the updated schema editor for API definitions
: The Definitions section of the API editor for the API Manager and API Designer UIs was updated in API Connect 10.0.6.0. However, the following limitations apply:

  - You cannot rename the top level schema.
  - The top level schema doesn't display the **`minProperties`**, **`maxProperties`**, and **`enum`** properties for the definition.
  - The **`OneOf`**, **`AllOf`**, and **`Enum`** schema structures aren't handled properly by the UI.

  You can work around these issues by editing the source YAML of your API document.

API Designer with Local Test Environment (LTE) fails to log in using https://localhost with the error message "Incorrect username, password, or credentials"

If you use the API Designer with the Local Test Environment and attempt to log in using the localhost, the login fails. You can work around the issue by configuring API Designer credentials to with the local host. Complete the following steps:

1. Download and extract API Designer, then install the credentials file as explained in steps 1, 2, and 6 in Installing the toolkit.
2. Edit the designer_credentials.json file and configure the following settings:
   - `"endpoint": "https://localhost"`
   - `"manager_endpoint": "https://localhost/manager"`
   - `"client_id": Client Id`
     The client ID displays on the console when you start the LTE platform-apic-lte. For more information, see Testing an API with the Local Test Environment.
   - `"client_secret": Client Secret`
     The client secret displays on the console when you start the LTE platform-apic-lte. For more information, see Testing an API with the Local Test Environment.
3. Start API Designer and log in with the LTE using https://localhost as the Host URL (the management endpoint).

Gateway timeout when validating very large API documents with the API governance service

If you are validating very large API documents (5 MB, or 100,000 lines, or more), with all of the rulesets selected by default, the API governance service is likely to take longer than 60 seconds to run, and this can result in an HTTP 504 gateway timeout. To workaround this issue, you can take the following actions:

- Validate the API document by using only one ruleset at a time.
- Increase your component deployment profile, for example to a three replica profile. For more information, see Deployment and component profiles, and Planning your deployment topology and profiles on VMware.

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

The IBM APIConnect operator uses leader election (leader-for-life) to ensure that only one operator/controller pod is running/active at a time. This relies on Kubernetes garbage collection to clean up the lock when the old operator pod is deleted. The new ibm-apiconnect operator pod fails to start if an existing terminating ibm-apiconnect operator pod is still holding the lock.
If you suspect you are encountering this issue, verify it by running the following command:

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

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

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

Resolve the issue by completing one of the following options:

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

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

  Kubernetes deletes the lock when the pod is deleted.

- Remove the lock manually by running the following command:

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

API Designer might hang while activating a large imported API

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

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

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

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

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

Permission restrictions in the API Designer UI

The API Designer UI currently has the following permission restrictions:

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

Deleted security requirements might remain in the API source

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

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

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

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

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

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

Where ***<site-alias>*** is the alias of your Developer Portal site.
To import the saved view after the upgrade, put the views.view.applications.yaml file inside a directory where it's the only file, and then run the following command:

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

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

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

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

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

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

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

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

If issues persist, contact IBM Support.

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

In the Test Preferences > Target gateway services setting, selecting Select compatible gateway services and choosing a specific gateway results in a "404 POST undefined" error when you test the API in the API Explorer or the Test tab.
Workaround: To avoid this problem, select the Use all compatible gateway services target gateway option instead.

Analytics command restrictions

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

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

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

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

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

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

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

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

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

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

Options menus in the Catalog might be hidden

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

Override plan rate limits are not displayed in the Endpoint tab

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

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

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

Consumer rate limit notifications are not available in Version 10

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

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

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

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

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

```
and requires database restart. accept restartDB to restart database]   Pending   10.0.1.5-3388-eus   10.0.1.5-3388-eus
2d6h
```

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

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

Performance impact when publishing products from API Designer

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

Limitations to the OpenAPI 3.0 support

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

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

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

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

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

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

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

If you set `enableOverride: false` to avoid overwriting indices during a restore, the restore operation should fail when existing indices are encountered. However, the operation does not fail, it instead gets stuck in the pending state.
When you configure the restore operation, set `enableOverride: true` but be aware that existing indices will be overwritten.

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

Options menu is missing from items on the Product list pages

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

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

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

Update WSDL option is not supported in the API Designer UI

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

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

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

- On a Kubernetes deployment: Troubleshooting upgrades on Kubernetes
- On an OpenShift deployment: Troubleshooting installation and upgrade on OpenShift
- On a VMware deployment: Troubleshooting upgrades on VMware

A unique email address is required for all Developer Portal accounts

Remember that every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a Developer Portal user account (and associated consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in to the Developer Portal: `A user already exists with this email address`.
For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries.

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

API Connect does not support a configured user registry that is based on a registry where `external_group_mapping_enabled` is set to `true`. Under certain conditions you can create a configured user registry based on a user registry with the unsupported option; if this happens you will encounter errors while onboarding users.
You can create and update user registries with the `apic user-registries:create` and `apic user-registries:update` commands. Both commands accept the `external_group_mapping_enabled` option. If you then create a configured user registry with the `configured_catalog_user_registries:create` command, the results depend on how you set the `external_group_mapping_enabled` option in the user registry.

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

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

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

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

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

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

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

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

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

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

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

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

An existing user cannot be added to a Space

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

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

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

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

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

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

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

With the DataPower API Gateway in IBM API Connect Version 2018.4.1, the type of object returned, for XML payloads, by an **`apim.getvariable('`*`context_name`*`.body')`** function call in a GatewayScript policy in an API assembly depends on how the variable is stored in the gateway context, as follows:

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

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

The problem does not occur with such an API migrated from IBM API Connect Version 5 because, with that release, **`apim.getvariable('`*`context_name`*`.body')`** also returns an XML Nodelist for XML payloads, nor does it occur if you are using the DataPower Gateway (v5 compatible).

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

Validate policy limitations when GraphQL response contains a GraphQL server error

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

A management backup remains in a **`Pending`** state

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

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

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

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

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

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

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

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

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

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

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

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

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

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

An OpenAPI definition that contains regular expression syntax fails validation

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

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

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

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

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

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

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

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

```
api_version: 2.0.0
name: name://../../../user-registries/<user_registry_name>/user-registry.yml
type: configured_api_user_registry
user_registry_url: file://../../../user-registries/<user_registry_name>/user-registry.yml
```

The migration of a user registry fails

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

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

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

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

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

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

and subsequently an error like the following:

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

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

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

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

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

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

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

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

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

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

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

4. Show the gateway peering status; for example:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Pagination setting is global across the API Connect user interfaces

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

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

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

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

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

Field validation for the Client security policy is incorrect

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

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

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

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

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

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

Adding members to a provider organization might fail

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

Upgrading with new certificates breaks analytics ingestion

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

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

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

User interfaces are not supported in Microsoft Edge

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

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

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

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

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

No option to bulk delete APIs or Products

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

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

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

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

Cannot publish an API that has duplicate security definition entries

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

Kubernetes might excessively evict pods

In Kubernetes, if insufficient memory, CPU, or storage resources are provided then pods are evicted randomly. To avoid this problem, allocate sufficient resources to the pods. In addition, there are known issues with log rotation in Kubernetes that are documented here: https://github.com/kubernetes/kubernetes/issues/59902. The lack of log rotation can cause logs to grow, limited only by the amount of storage on the node, eventually causing pods to restart. Therefore ensure that appropriate log levels are configured for all API Connect components.
You may also follow the instructions described here https://success.docker.com/article/how-to-setup-log-rotation-post-installation to set the maximum size of the log before it is rolled (max-size) and the maximum number of log files that can be present (max-files). More information about the Docker JSON File logging driver is available here: https://docs.docker.com/config/containers/logging/json-file/.

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

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

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

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

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

Redact conditions within **`switch`**, **`operation-switch`**, or **`if`** policies might not execute after converting to DataPower API Gateway
: If an API Connect v5-compatible **`redact`** policy is found within a **`switch`**, **`operation-switch`**, or **`if`** policy, the migration utility does not move the **`redact`** policy to the beginning or end of the assembly. The difference in the response between API Connect v5 and DataPower API Gateway may prevent data from being redacted in the DataPower API Gateway.
For example, if an assembly includes a **`switch`** policy containing four redact conditions followed by an **`invoke`** policy, each redact condition redacts the response data. After porting to the API Gateway, the redact conditions remain inside the **`switch`** policy and target the **`message.body`** property for redaction. These redactions fail to execute because the **`message.body`** property has not yet been retrieved by the **`invoke`** policy. To correct this problem, you must move the **`invoke`** policy before the **`switch`** policy in the assembly.

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

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

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

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

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

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

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

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

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

API Connect concepts

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

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

Who does what in API Connect?

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

Systems administrators

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

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

- Cloud owner: There is one cloud owner for a deployment, and that person adds other people to the server management team.

- Cloud administrator: Your deployment needs at least one cloud administrator; your company will probably assign multiple people to perform this role. Cloud administrators manage the deployment and configuration of API Connect servers and software.

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

Users

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

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

Provider organization
: A provider organization performs many tasks during the lifecycle of an API, from developing, publishing, and maintaining APIs to managing the membership of the organization itself. Each p-org can be as large or as small as needed. A large company might create a p-org for each product team, or for each department but a small company might use a single p-org.
A person can be a member of multiple provider organizations and receive a different set of permissions for each. When the user logs in to API Connect, they select the organization whose resources they will use, and receive the appropriate permissions for that organization for the duration of the session.

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

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

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

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

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

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

API Connect user roles

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

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

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

- User roles in the Cloud Manager UI.
- User roles in the API Manager UI.
- User roles in the Developer Portal UI.

User roles and permissions in the Cloud Manager UI

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

Table 1. Cloud Manager UI permissions

Permission	Action	Meaning
Cloud Settings	View	View all items on the Cloud Manager > Settings menu (except Roles)
	Manage	Add, update, and delete all items on the Cloud Manager > Settings menu (except Roles)
Member	View	View members on the members list located at Cloud Manager > Members

	Manage	Add and invite members from Cloud Manager > Members
		Note: By default, a user with Member > Manage permission can assign, to themselves or to another user, any role with any permission regardless of the permissions that they themselves have. However, you can apply a restriction such that, for a user to assign a role, they must themselves have at least all of the permissions that are applied to that role. To apply that restriction, complete the following steps:
		1. Log in to your management server from the toolkit CLI as a member of the cloud administration organization; for details, see Logging in to management server.
		2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):
		`apic cloud-settings:update --server mgmt_endpoint_url -`
		where *mgmt_endpoint_url* is the platform API endpoint URL.
		3. Enter the following data, followed by a new line:
		`restrict_member_manage_permission: true`
		4. Press `CTRL D` to terminate the input.

Org	View	Org:View is a permission assigned to all Roles in Cloud Manager. It does not provide access to any functionality. It allows a user to activate their membership. It is the only permission in the Member role.
Provider-Org	View	View the list of provider organizations at Cloud Manager > Provider Organizations
	Manage	Add, edit, and delete provider organizations and invite owners from Cloud Manager > Provider Organizations
Settings	View	View the items on the Cloud Manager > Resources menu plus Roles located at Cloud Manager > Settings > Roles
	Manage	Add, edit, and delete the items on the Cloud Manager > Resources menu plus Roles located at Cloud Manager > Settings > Roles
Topology	View	View the items on the Cloud Manager > Topology menu
	Manage	Add, edit, and delete the items on the Cloud Manager > Topology menu
Analytics	View	View items on Cloud Manager > Analytics, and also create, update, duplicate, delete, share, and unshare saved queries.

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

Table 2. Cloud Manager UI roles

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

User roles and permissions in the API Manager UI

The following tables describe the API Manager UI user permissions.

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

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

Table 3. Organization permissions

Permissions	Action	Permits the member to
Member	View	View organization's members

Permissions	Action	Permits the member to

	Manage	Manage organization's members
		Note: By default, a user with Member > Manage permission can assign, to themselves or to another user, any role with any permission regardless of the permissions that they themselves have. However, you can apply a restriction such that, for a user to assign a role, they must themselves have at least all of the permissions that are applied to that role. To apply that restriction, complete the following steps:
		1. Log in to your management server from the toolkit CLI as a member of the cloud administration organization; for details, see Logging in to management server.
		2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):
		`apic cloud-settings:update --server mgmt_endpoint_url -`
		where *mgmt_endpoint_url* is the platform API endpoint URL.
		3. Enter the following data, followed by a new line:
		`restrict_member_manage_permission: true`
		4. Press **CTRL D** to terminate the input.

Settings	View	View an organization's configuration settings, including roles, TLS profiles, and user registries. View configuration settings for a Catalog or Space, including policies and OpenAPI extensions.
	Manage	Manage an organization's configuration settings, including roles, TLS profiles, and user registries. Manage configuration settings for a Catalog or Space, including policies and OpenAPI extensions.
Topology	View	Same permissions as **`Settings: View`**.
	Manage	Same permissions as **`Settings: Manage`**.
Org	View	View an organization
Product-Drafts	View	View draft APIs and Products
	Edit	View draft APIs and edit draft Products
Api-Drafts	View	View draft APIs
	Edit	Edit draft APIs and view draft Products
Product	View	View Products
	Stage	Stage Product
	Manage	Manage Product
Product-Approval	View	View Product lifecycle changes
	Stage	Approve the staging of a Product
	Publish	Approve the publishing of a Product
	Supersede	Approve the superseding of a Product
	Replace	Approve the replacement a Product
	Deprecate	Approve the deprecation of a Product
	Retire	Approve the retiring of a Product
Consumer-Org	View	View consumer organization and developers
	Manage	Manage consumer organization and developers
App	View	View both production and development applications.
	Manage	Manage both production and development applications. A member with this permission can also request the promotion of a development app to a production app. This request triggers a task that needs approval by a member with the App-approval Manage permission.
App-Dev	Manage	Same permissions as **`Settings: Manage`**.
App-Approval	View	View application approvals, for requests to promote a development app to a production app.
	Manage	Manage (Approve or Decline) requests for approval to promote a development app to a production app.
Subscription	View	View application Plan subscriptions that have been created by application developers in the Developer Portal.
	Manage	Manage the application Plan subscriptions that have been created by application developers in the Developer Portal. The Manage permission includes ability to migrate a subscription to another plan.
Subscription-Approval	View	View application Plan subscription approvals.
	Manage	Manage (approve or decline) application Plan subscriptions.
Consumer-Onboard-Approval	View	View consumer onboard approvals.
	Manage	Manage (approve or decline) consumer onboard approvals.
Api-Analytics	View	View analytics data, as well as access and apply saved analytics queries.
	Manage	In addition to the view permissions, the user can create, update, duplicate, delete, share, and unshare saved analytics queries.
Child	View	At the provider organization level, view Catalogs in the provider organization. At the Catalog level, view Spaces in the Catalog.
	Create	At the provider organization level, create Catalogs in the provider organization. At the Catalog level, create Spaces in the Catalog.
	Manage	At the provider organization level, manage Catalogs in the provider organization. At the catalog level, manage Spaces in the Catalog. Management tasks including deleting a Catalog or Space, or transferring ownership of a Catalog or Space.

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

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

Role	Role description	Permissions	Actions

Role	Role description	Permissions	Actions
Organization Owner	A provider organization owner has the full set of access permissions to API Connect functions, and also commission APIs and tracks their business adoption.	All permissions	All actions.
Administrator	A provider organization administrator has, by default, the full set of access permissions to API Connect functions, and also commission APIs and tracks their business adoption.	All permissions	All actions.
API Administrator	API administrators manage the lifecycle of APIs and publish APIs for discovery and use.	All permissions	All actions except cannot manage the following permissions: Member, Settings, Topology, and Child.
Community Manager	A community manager manages the relationship between the provider organization and application developers, provides information about API usage, and provides support to application developers.	Member	View
		Settings	View
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View, Edit
		Product	View
		Product-approval	View
		Consumer-org	View, Manage
		App	View, Manage
		App-dev	Manage
		App-approval	View, Manage
		Subscription	View, Manage
		Subscription-approval	View, Manage
		Consumer-onboard-approval	View, Manage
		Api-analytics	View, Manage
		Child	View
Developer	API developers design and develop APIs and applications for the provider organizations to which they belong. Note: The Developer role allows the creation of Products and APIs, and the staging and publishing of Products to a Catalog or Space, when assigned to a user at the provider organization level--but not when assigned to a user who is a member only of a Catalog or Space within a provider organization. A Developer in a Catalog or Space can manage Products that are staged or published to the Catalog or Space.	Member	View
		Settings	View
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View, Edit
		Product	View, Stage, Manage
		Product-approval	View, Stage, Publish, Supersede, Replace, Deprecate, Retire
		Consumer-org	View
		App	View, Manage
		App-dev	Manage
		App-approval	View, Manage
		Subscription	View, Manage
		Subscription-approval	View, Manage
		Api-analytics	View, Manage
		Child	View, Create
Member	Member of a provider organization	Org	View
Viewer	Viewer of a provider organization	Member	View

Role	Role description	Permissions	Actions
		Topology	View gateway services or portal services at the provider organization.
		Org	View
		Drafts	View
		Product-approval	View
		Consumer-org	View
		App	View
		App-approval	View
		Subscription	View
		Subscription-approval	View
		Api-analytics	View
		Child	View

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

User roles in the Developer Portal UI

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

Table 5. Developer Portal UI roles

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

Role	Role Description	Permission	Actions
		Consumer app-analytics	View
Member	Member of the app developer organization	Organization	View
Viewer	Viewer of the app developer organization	Organization member	View
		Organization settings	View
		Organization	View
		Consumer product	View
		Consumer app	View applications
		Consumer production-app	View production applications
		Consumer app-analytics	View application analytics

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

Packaging strategy and terminology in API Connect

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

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

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

- APIs
- Plans and Products
- Catalog and Spaces
- Organizations and users
- LoopBack® applications
- Sample provider organization with two Catalogs

APIs

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

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

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

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

Plans and Products

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

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

Plans perform the following functions:

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

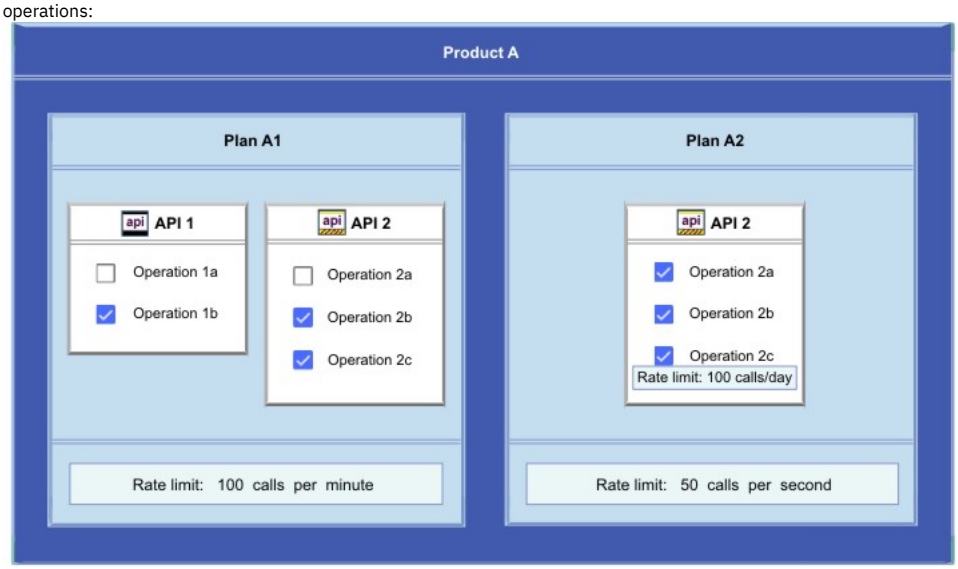
A rate limit can be implemented as a default rate that is shared across all operations in a Plan, or can be set for specific operations of an API. Plans can use differing rate limits to provide different levels of service to API consumers. For example, a "Demo Plan" might enforce a rate limit of ten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

permit up to 1000 calls per second.

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

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

Multiple Plans within a single Product provide different levels of performance for the same offering. For example, a Product can include a "Demo Plan" that makes a single API available, and a "Full Plan" that makes several APIs available.
The following diagram illustrates how Plans can be used to make operations from one or more APIs available, and to set rate limits on both Plans and individual operations:

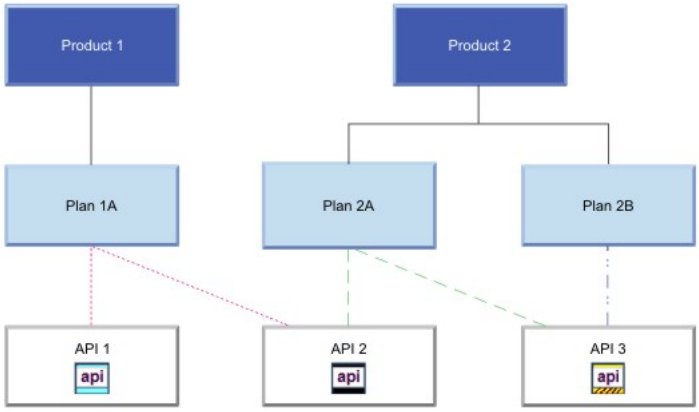


The diagram illustrates the following concepts:

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

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

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



For more information about managing the lifecycle of Products, see Working with Products in the API Manager.

Catalogs and Spaces

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

API Connect includes a syndication feature that enables API providers to partition a Catalog into multiple staging targets (or *Spaces*) for API development purposes. Each API provider development team can use its own dedicated Space to manage its Products independently of other teams. A Space has its own set of capabilities relating specifically to the Products and APIs that are created and published to that Space. Products and APIs in all Spaces in a given Catalog are published to the same Developer

Portal. Spaces are not visible on the Developer Portal. Application developers who consume the APIs on the Developer Portal are unaware of the Space configuration used by the API Developers. On the Developer Portal, the APIs are seen as a coordinated offering within a Catalog.

For more information about Catalogs and Spaces, see Working with Catalogs and Using syndication to partition Catalogs into Spaces.

Organizations and users

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

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

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

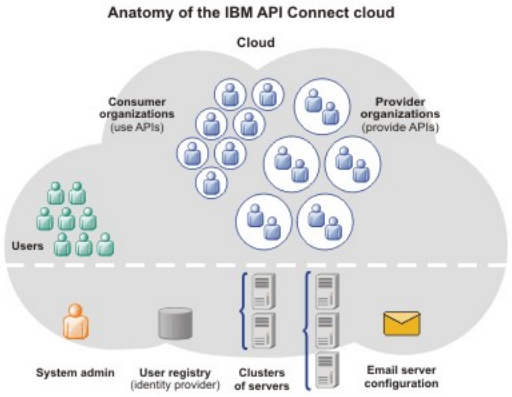
A consumer organization owns only *developer* applications, and consumes the APIs and applications produced by the provider organization. Standard responsibilities within a consumer organization include:

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

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

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

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



For more information about organizations, see Administering provider organizations and Administering consumer organizations.

LoopBack applications

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

For information about creating LoopBack applications, see Creating APIs and applications.

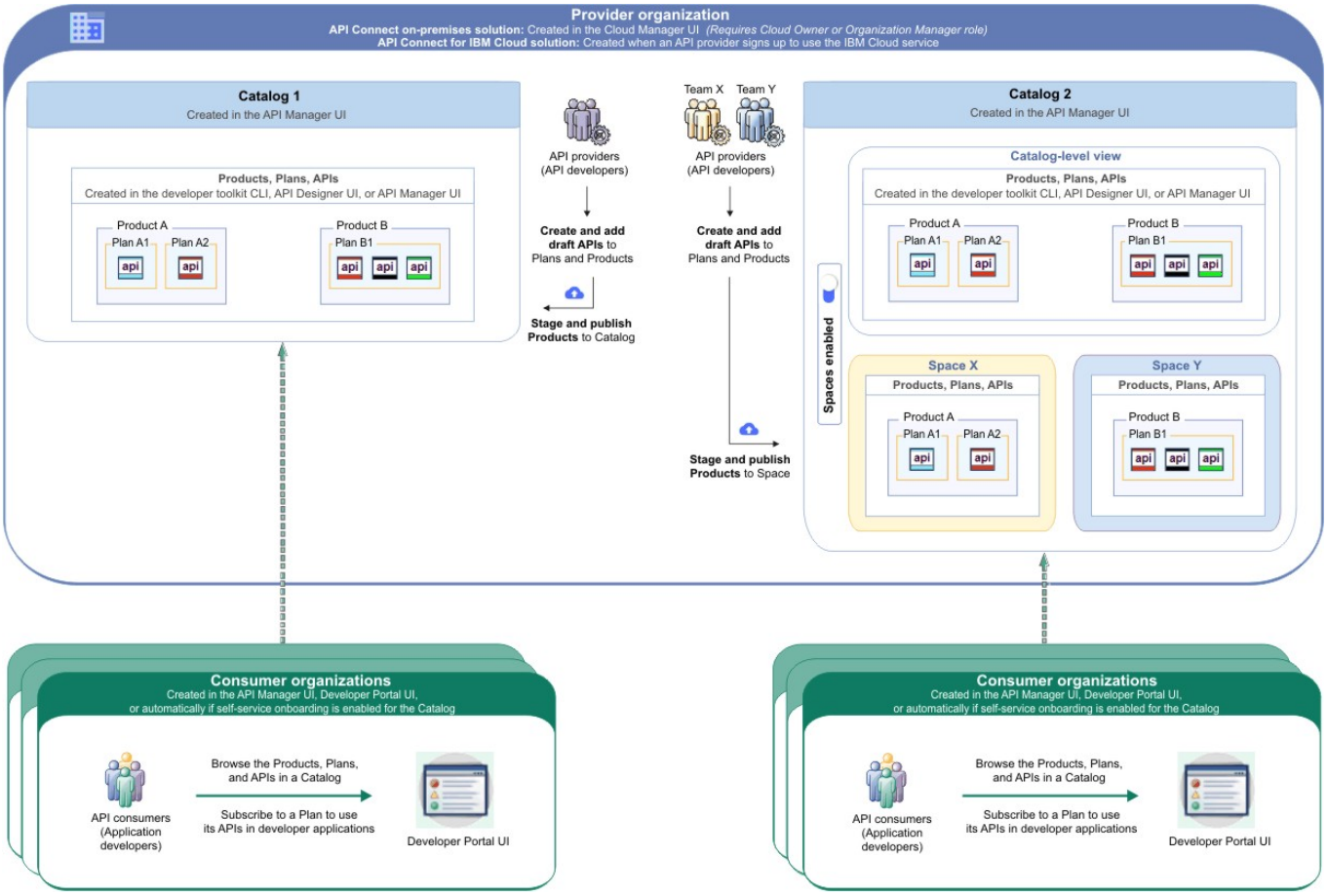
Sample provider organization with two Catalogs

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

- Catalog 1 does not require separation of the Products for use by individual provider development teams, and so does not have Spaces enabled. API developers with access to this Catalog create draft APIs, and stage and publish them as Products to the Catalog. Several consumer organizations are granted permission to explore,

discover, and use the published Products and APIs. In each of the consumer organizations, the published Products and APIs from Catalog 1 are exposed on a single Developer Portal.

- Catalog 2 is partitioned into two Spaces (X and Y) so that two provider development teams can manage their Products independently. These teams stage and publish their APIs (as Products) separately to the individual Spaces, to make them accessible to application developers in multiple consumer organizations. In each of the developer organizations, the published Products and APIs from both Spaces are exposed in the same Developer Portal, and application developers who access this portal will see the APIs from both Spaces as a coordinated offering.



- **Understanding rate limits for APIs and Plans**
  In API Connect, you can configure rate limiting on APIs and Plans to manage network traffic and API usage.

Understanding rate limits for APIs and Plans

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

What are rate limits and burst limits?

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

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

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

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

Where can you apply rate limits?

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

The following list explains where you can configure rate limits:

Plan: all APIs
: You can define a general rate limit at the Plan level, and it affects all of the consumer orgs that use that Plan. When a rate limit is enforced for a Plan, the limit is checked before every call that involves any API contained in that Plan. If the rate limit is exceeded, APIs are not even called. For information on setting a rate limit for a Plan, see Editing a draft Product.
  Note: If the subscriber changes to a different Plan, only the Plan-level rates limit are reset. If there is a hard rate limit specified in the API's definition or assembly, that rate limit is still enforced under the new Plan.

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

  - Defining rate limits for an API operation
  - Describing Plans in your Product

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

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

  - Rate Limit policy
  - GatewayScript policy
  - Using context variables in a GatewayScript policy

How can you test rate limits?

When you're ready to test rate limits on an API or Plan, execute a larger number of calls than specified by your rate limit, within a shorter period of time. If you set soft rate limits, check for warning messages that indicate when each limit is exceeded. If you set hard limits, you can tell it was enforced when the API call fails. You can also check the error messages to verify which calls exceeded the limits.

For more information on creating and configuring Catalogs, see Working with Catalogs.

API Connect components

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

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

Cloud Manager

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

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

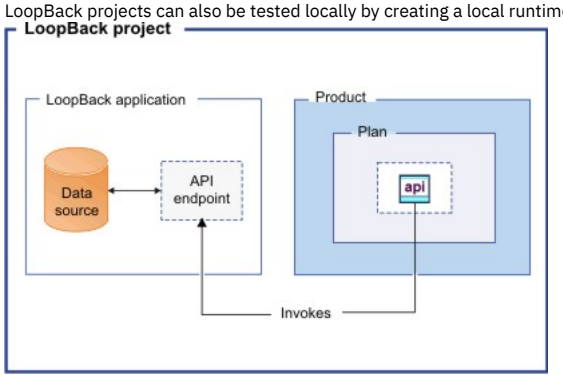
For more information about the Cloud Manager, see Managing your cloud.

The developer toolkit

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

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

Using LoopBack, an API developer can create a Node.js application, connect to a data source such as a back-end database or a REST API to be consumed, and then expose the application as a REST API by creating a model definition. A LoopBack model defines the application data, validation rules, data access capabilities, and business logic for an API, and provides a REST API by default. This REST API can then be used by a REST API definition that was created using the API Designer or CLI and exposed to your users. The API and its associated application, which are implemented as a LoopBack project, must both be published to enable the project to be run. LoopBack projects can also be tested locally by creating a local runtime environment. The following diagram illustrates the LoopBack project architecture:



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

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

API Manager

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

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

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

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

For more information about the API Manager, see Managing your APIs.

API Gateways

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

API Connect supports the following types of API gateways:

- DataPower Gateway (v5-compatible)
  The DataPower® Gateway is an enterprise API Gateway that is built for departments and cross-enterprise usage. In V10 API Connect, the DataPower Gateway is compatible with the V5 DataPower Gateway, so it is referred to in documentation as the "v5-compatible" (or "v5c") gateway. The v5-compatible gateway provides a comprehensive set of API policies for security, traffic management, mediation, acceleration, and non-HTTP protocol support.
- DataPower API Gateway
  The DataPower API Gateway was designed with the same security focus as the DataPower Gateway (v5-compatible). Where DataPower Gateway (v5 compatible) is built for flexibility, DataPower API Gateway is built specifically for the API use case, with resulting performance benefits.DataPower API Gateway is optimized for the cloud. Use this gateway if you are running applications in a public or private cloud and want to expose them as APIs.

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

For more information about the gateways, and a comparison of their features, see Gateway types.

Runtime

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

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

Developer Portal

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

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

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

API Analytics

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

API Connect server requirements

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

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

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

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

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

Typical tasks per interface component

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

Table 1. API Connect Tasks per interface component

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

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

- **Gateway types**
  IBM API Connect provides the different types of gateways for use with your deployment.

Related concepts

- Packaging strategy and terminology in API Connect

Gateway types

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

DataPower API Gateway

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

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

DataPower Gateway (v5 compatible)

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

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

Gateway comparison

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

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

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

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

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

API Connect support

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

For the most recent API Connect technical information, see the IBM Support Community.

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

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

API Connect glossary

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

API Administrator (role)
: Manages the API product lifecycle for the Provider Organizations for which they are a member.

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

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

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

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

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

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

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

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

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

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

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

client by using an HTTP query parameter.

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

Cloud Administrator (role)
: Manages the configuration of resources, regions, and availability zones for the Admin Organization of an on-premises installation.

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

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

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

Community Manager (role)
: Manages application developer communities for the Provider Organizations for which they are a member.

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

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

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

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

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

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

Gateway
: See API Gateway.

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

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

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

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

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

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

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

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

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

Management service
: Consists of one or more Management servers.

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

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

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

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

Notification services
: Integrations that provide notification capabilities, such as email.

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

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

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

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

Organization Manager (role)
: The Organization Manager manages Provider Organizations for the Admin Organization.

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

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

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

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

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

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

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

Resources (User registries, TLS Profiles, Notifications)
: Resources supply necessary functions for the API Connect cloud, such as user authentication, SSL security, and sending system-generated emails.

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

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

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

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

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

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

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

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

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

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

Toolkit
: See Developer Toolkit.

Topology administrator (role)
: Configures gateway services for the Admin Organization.

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

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

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

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

Accessibility features for IBM API Connect

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

Accessibility features

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

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

Tip: This product documentation, and its related publications, are accessibility-enabled for the IBM Home Page Reader. You can operate all features by using the keyboard instead of the mouse.
If you are reading a PDF file with a screen reader, the default reading option typically returns the best results. In some cases, how the PDF file is generated might require you to select one of the other reading options. For example, Use reading order in raw print stream.

Keyboard navigation

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

For more information about the commitments that IBM makes towards accessibility, see the IBM Accessibility Center.

Legal information

Notices, and terms and conditions for information centers.

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

Tracking API volume for auditing and compliance

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

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

Usage archiving

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

Overage charging

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

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

Notices

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

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

Trademarks

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

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

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

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

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

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

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

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

Privacy Policy Considerations

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

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

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

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

Terms and conditions for information centers

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

Applicability

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

Personal use

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

Commercial use

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

Rights

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

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

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

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

IBM API Connect Considerations for GDPR Readiness

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

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

Notice:

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

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations.

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

Table of Contents

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

GDPR

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

Why is GDPR important?

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

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

Read more about GDPR

- Complete guide to GDPR compliance
- IBM GDPR website

Product Configuration - considerations for GDPR Readiness

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

Configuration to support data handling requirements

The GDPR legislation requires that personal data is strictly controlled and that the integrity of the data is maintained. This requires the data to be secured against loss through system failure and also through unauthorized access or via theft of computer equipment or storage media.
IBM API Connect stores identity data in a local database. This encompasses both clients' employee identity data and end users' identity data. Direct access to this database is not available. Identity information collected is protected in transit; refer to Creating a TLS Server Profile for details on configuring TLS profiles.

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

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

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

Configuration to support Data Privacy

For Developer Portal, you can customize the privacy policy statement. Refer to Customizing the privacy policy statement for details.

Configuration to support Data Security

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

Data Life Cycle

GDPR requires that personal data is:

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

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

User account information

API Connect collects and stores identity information, including first and last name, and email address, for the purposes of user registration. Cloud Manager and API Manager accounts are for your employees (or designated actors). Developer Portal accounts are for your consumers of your APIs. Identity information can be collected directly from users or can be copied from LDAP registries. In situations where non-local user registries are used, only the email address is copied from the LDAP registry. Developer Portal user accounts can be deleted; refer to Deleting your Developer account for details. Cloud Manager and API Manager user accounts' identity information can be anonymized by users.
Users of the API Manager UI can publish Products and APIs to the Developer Portal for Application Developers to access and use. Refer to Developer Portal: socialize your APIs to learn about Developer Portal. Developer Portal accounts are for consumers of your APIs. You can define and customize a terms and conditions statement that your users must accept before they can register to use your Developer Portal; refer to Customizing the terms of use statement for details.

API analytics

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

The retention period for storing Analytics data in API Connect is configurable; refer to Retention and rollover for details. Backup capability for this information is not available.

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

System logs

Event-logging preferences can be configured at the API level, refer to Activity Log for details.

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

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

Data Collection

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

Types of Data Collected

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

Data Storage

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

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

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

Data Access

Identity information can be viewed by administrators that you define.

Analytics information can be accessed through a variety of means. Refer to Access your analytics data for details.

Analytics information can be offloaded to third-party systems. Refer to Offloading to third-party systems for details. It is your responsibility to protect off-loaded data.

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

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

Data Processing

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

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

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

Data Deletion

Right to Erasure

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

Data Deletion characteristics

Users can delete their own Developer Portal user accounts -- refer to Deleting your Developer account for details. Cloud Manager and API Manager user account identity data can be anonymized by the users to remove association with the account data.
Technical information related to service use collected in logs is rolled over based on size and time criteria.

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

The retention period for storing Analytics data in API Connect is configurable; refer to Retention and rollover for details. Backup capability for this information is not available.

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

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

Data Monitoring

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

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

Capability for Restricting Use of Personal Data

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

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

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

Essential reading

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

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

Creating A Digital Ecosystem – Past, Present, and Future
: The ability to create a digital ecosystem is critical to digital transformation success. Your success is your network reach.

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

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

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

Providing APIs or Managing APIs – There is a Big Difference
: A discussion of the potential for confusion between APIs that are provided and APIs that are managed.

Recommendations for an API Economy Center of Excellence
: What roles are required to drive a successful API initiative, how should this fit in the current organization, and how do these roles relate to existing roles in the company?

Focus on the API Developer
: A productive developer is a happy developer. One of the most frequently discussed topics in the API economy is focusing on the needs of the Application developer – the consumer target for your APIs.

Agile API development
: Agile API Development Customer expectations and behavior are continuously changing. To deliver exceptional customer experience, a business must be nimble to adapt to these changing needs.

API Products – Who, What, Where, When, Why and How”
: An API Product is an API offering made available for consumer use that is offered to a target market to satisfy a customer’s needs.

Changing Culture – How Committed Are You?
: How do we change the culture in our organization to create an API culture?

Plan Ahead! Don’t Build an API Superhighway into a Cul-de-sac
: Without proper planning, a business can start their API initiatives, build incredible excitement quickly, but find that the path they have taken leads them into a cul-de-sac (or dead end) that cannot handle the demand they have created.

IBM API Connect v10.x Deployment WhitePaper
: A technical deep dive on the deployment options for IBM API Connect.

Principles for API Security - White Paper
: API security is of paramount importance in gaining the promised benefits without exposure to negative consequences.

Can you trust your APIs?
: As enterprises are continuously expanding their digital footprint, they must ensure the API behavior is intact, as it has a far-reaching effect on an application's execution and end-user experience.

Istio Service Mesh and APIConnect/DataPower Gateway integration
: What is Istio, and how can DataPower API Gateway integrate in an Istio Service Mesh.

Best Practice – Monetizing API Products
: Ultimately the goal for almost all API initiatives is Monetization in the broad sense that crosses multiple modes from Free - to Indirect - to Direct. Most businesses using APIs successfully are monetizing APIs by speeding offerings to market, reaching new markets and new customers, innovating, and sharing assets better inside their enterprise.

Tutorials

These tutorials guide you through the steps typically required to set up, configure and run an API Connect cloud.

Introduction

These tutorials are arranged according to the desired goal, such as creating and publishing an API, or managing catalogs of products, or consuming published API products. These goals can be grouped into categories as follows.

Table 1. Tutorial Categories

Category	Description
Cloud Administration	Setting up and configuring the API Connect cloud, including the creation of Provider Organizations.
API Provider Administration	Setting up and managing catalogs, inviting organization members (such as developers), creating Developer Portals, managing resources.
API Product Promotion and Usage	Managing the Developer Portal presentation and user administration, as well as consuming published API products as an external developer.

Time required

Each tutorial should take approximately 20 - 30 minutes to finish. If you explore other concepts that are related to this tutorial, it might take longer.

Use the links in the following sections to access the tutorials that fit your goals.

Cloud Administration

Table 2. Cloud Administration Tasks. Setting up and configuring the API Connect cloud, including the creation of Provider Organizations.

Category	Tutorials
Initial Configuration	• Initial Cloud Configuration
Provider Organization	• Creating a Provider Organization

API Provider Administration

Table 3. API Provider Tasks. Setting up and managing catalogs, inviting organization members (such as developers), creating Developer Portals, managing resources.

Category	Tutorials
Developer Portal Set Up	• Creating the Portal • Register a Consumer Application

API Product Promotion and Usage through the Developer Portal

There are many tutorials available for the Developer Portal from getting started to advanced customization. For more information, see Developer Portal tutorials.

Prerequisites

- You must have a web browser available, whether you are working online or offline.
- When publishing Products in any of the tutorials you must have the permissions of an Organization Manager or Administrator. However, you can complete several of the tutorials with fewer permissions. For more information about user roles, see Administering user access.

Installing and maintaining IBM API Connect

To ensure that your IBM® API Connect cloud functions, your cloud must have the necessary system requirements to support the installation. During the installation process, the components of IBM API Connect can be configured to satisfy your requirements.

Note: For a comprehensive technical guide to best practices, considerations, and deployment options for API Connect, see the IBM API Connect v10.x Deployment WhitePaper .

- **Kubernetes, OpenShift, and Cloud Pak for Integration**
  Use the instructions in this section to install, upgrade, and maintain API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.
- **VMware**
  Use the instructions in this section to install, upgrade, and maintain API Connect on VMware.
- **Migrating from IBM API Connect v5 Public Cloud to v10.0.6x**
  You cannot migrate direct to IBM API Connect 10.0.6.0 or later from API Connect v5 Public Cloud. You must migrate to version 10.0.5.x or later, and then upgrade to 10.0.6.0 or later.
- **Migrating a version 5 deployment to version 10.0.x**
  See the following note for information about migrating from API Connect 5.0.x.
- **Migrating v5-compatible APIs to API Gateway**
  Use the migration utility to automate the migration of v5-compatible APIs to DataPower API Gateway APIs.

- **Migrating from v10 to v10 on a different form factor**
  Migrate an API Connect v10 deployment from one form factor to a different form factor.
- **[Technical Preview] Using Config Sync to replicate consumer-side catalog data**
  Use the API Connect Config Sync utility to replicate consumer-side data from a source catalog in an API Connect cluster to a target catalog in a different API Connect cluster, so that API calls can be served by either API Connect cluster interchangeably to enable hybrid deployments and high-availability style scenarios.
- **The API Connect operations tool: apicops**
  Use the apicops operations tool to check the health, troubleshoot, and run maintenance operations on your API Connect installation.

Kubernetes, OpenShift, and Cloud Pak for Integration

Use the instructions in this section to install, upgrade, and maintain API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

- **Planning your deployment**
  Review the following topics to plan your API Connect deployment.
- **Installing API Connect**
  Use these instructions to install a deployment of API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.
- **Upgrading API Connect**
  Upgrade API Connect on Kubernetes and OpenShift platforms to a newer version.
- **Maintaining API Connect**
  You can use utilities to complete maintenance tasks such as backup, restore, and certificate management on Kubernetes.
- **Uninstalling API Connect**
  You can uninstall a Kubernetes deployment of API Connect.

Planning your deployment

Review the following topics to plan your API Connect deployment.

- **Prerequisite skills for administering API Connect**
  The background knowledge and skills that are required to install and manage an API Connect deployment.
- **Key concepts: Custom Resources, Operators, and Operands**
  API Connect uses Custom Resources (CRs) and the Kubernetes Operator pattern.
- **Deployment profiles, topologies, and strategies**
  Review the available deployment topologies, profiles, and strategies for high-availability and disaster recovery, and choose the best option for your organization.
- **Kubernetes ingress controller prerequisites**
  Describes the prerequisite settings for the ingress controller for a Kubernetes runtime environment.
- **Persistent Volume (PV) requirements**
  Review the persistent volumes (PVs) required by your API Connect installation.
- **Load balancer configuration in a Kubernetes deployment**
  When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect Kubernetes deployment.
- **Firewall requirements on Kubernetes**
  Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on Kubernetes.
- **Planning your analytics deployment**
  Plan your API Connect analytics deployment by reviewing options for deployment profile and estimating required persistent storage space.
- **Key Concepts: Cert-manager, Issuers, and Secrets**
  By default API Connect uses an open source product that is called cert-manager to handle the issuing and renewal of the certificates that are used by API Connect. The cert-manager has its own Kubernetes pods and runs in its own namespace. The cert-manager adds some additional resources to the Kubernetes environment. The API Connect administrator needs to be familiar with these additional resources:
- **Certificates in a Kubernetes environment**
  Use of cert-manager is recommended for managing certificates in a Kubernetes environment.
- **Webhooks deployed and managed by the API Connect Operator**
  The API Connect Operator deploys and manages various Kubernetes API webhooks in the cluster to assist in management of its custom resources.
- **Administering API Connect in your container environment**
  How to manage your API Connect subsystems in Kubernetes and OpenShift environments.

Prerequisite skills for administering API Connect

The background knowledge and skills that are required to install and manage an API Connect deployment.

The following list provides an idea of the concepts and technologies that the API Connect administrator should be familiar with:

- The container platform where API Connect is to be installed, either:
  - Kubernetes
  - OpenShift

  How to manage pods, PVCs, namespaces in your container platform.
- Linux: File permissions, editing files with the vi editor, redirecting command output to file.
- Network concepts: FQDNs, ports, firewalls.
- TLS: CAs, certificate chains, mTLS.

The examples in this list do not include all background knowledge that is required, but are intended to give a general idea of the skills that are needed to plan, install, and maintain an API Connect deployment.

Key concepts: Custom Resources, Operators, and Operands

API Connect uses Custom Resources (CRs) and the Kubernetes Operator pattern.

- Kubernetes: https://kubernetes.io/docs/concepts/extend-kubernetes/operator/
- OpenShift: https://developers.redhat.com/articles/2021/06/11/kubernetes-operators-101-part-1-overview-and-key-features.

Below are some key points to understand about how API Connect implements this pattern:

- An API Connect instance is composed of two main operators and the API Connect subsystems that they manage, which in Kubernetes terminology are known as the Operands.
- The operator is responsible for monitoring the API Connect operands, ensuring that they are running as expected.
- API Connect operands can continue running without their operator, in an unmanaged state. However, this is not a supported configuration.
- The API Connect operator manages the Management Server, Portal, and Analytics subsystems. On Kubernetes each of these subsystems has its own Custom Resource, so each of these is an Operand. On Openshift and Cloud Pak for Integration there is also the option of having a top-level Custom Resource which manages all subsystems as a single Operand.
- The DataPower operator manages the Gateway operand.
- A single API Connect operator can manage multiple API Connect instances if the operator is installed globally and the API Connect operands are in individual namespaces. This is the recommended configuration where multiple API Connect instances are required, as opposed to a separate operator in each namespace.
- There can be only one instance of the API Connect operator in your Kubernetes or OpenShift environment. You cannot have multiple namespaces each with their own API Connect operator.
- In a single OpenShift/K8S environment all API Connect instances, operands, and operators, in all namespaces must be at the same version. The only exception is during upgrade where the operators are upgraded first.

Deployment profiles, topologies, and strategies

Review the available deployment topologies, profiles, and strategies for high-availability and disaster recovery, and choose the best option for your organization.

- **Deployment and component profiles**
  Decide on whether to deploy three replica or one replica API Connect subsystems, and the memory and CPU resources to allocate to the subsystems.
- **Multiple data center deployment strategies**
  Strategies for deploying API Connect across multiple data centers.
- **OpenShift: Deciding to use individual CRs, a top-level CR, or Cloud Pak for Integration**
  You can install API Connect subsystems separately by using individual subsystem CRs, or you can deploy them all together with a top-level CR. You can also deploy API Connect as part of Cloud Pak for Integration. The differences between each installation type are described in this topic.

Deployment and component profiles

Decide on whether to deploy three replica or one replica API Connect subsystems, and the memory and CPU resources to allocate to the subsystems.

API Connect is composed of three components: Management, Portal, and Analytics, and a supporting program: the DataPower Gateway, see API Connect components.
Note: Throughout this document, the DataPower Gateway supporting program is also referred to as a "component" or a "subsystem". The Management, Portal, and Analytics components are also referred to as "subsystems".
The Management, Portal, Analytics, and Gateway each have a set of capacity and HA profiles, called component profiles. OpenShift and Cloud Pak for Integration also have API Connect deployment profiles that define the component profiles that are used by each component of a complete API Connect deployment.

The naming convention of the profiles is `nAxcB.mC` where:

- **nA** indicates the number (A) of replicas of each pod, where each replica must run on a different worker node.
- **cB** indicates the minimum number of virtual cores (B) required on the worker node to deploy the profile.
- **mC** indicates the minimum amount of memory (C) in GB required by the profile on the worker node.

Component profiles

Each API Connect component has a defined set of profiles that it can be deployed with. For information on the resource settings defined for each component profile, see Component profiles, CPU limits, memory limits, and licensing.

Key points of component profiles:

- One replica profiles are denoted by `n1`, and provide a single replica of each pod. The one replica profiles can be deployed on a single worker node.
- Three replica profiles are denoted by `n3`, and provide three replicas of each pod. At least three worker nodes must be available to deploy a three replica profile. If one replica pod fails, then the other two replicas continue serving requests, providing high availability.
- Component profiles do not have to be the same for each component. For example, you can have a three replica deployment of the Management component that is configured with a one replica deployment of the Portal component.
- It is possible to change component profile after installation see Advanced configuration.
- Multiple Portal, Analytics, and Gateway component instances can be added with either configuration. For example, you can have a three replica Management component that is configured with two Portal services: a one replica Portal component, and a three replica Portal component.

OpenShift and Cloud Pak for Integration API Connect deployment profiles

API Connect deployment profiles define a complete API Connect deployment, specifying the component profiles for the Management, Portal, Analytics, and Gateway components to be used in that deployment.

Key points of API Connect deployment profiles:

- The API Connect deployment profile is selected by the user in the installation steps on OpenShift and Cloud Pak for Integration.
- API Connect deployment profiles provide one instance of the Portal, Analytics, and Gateway components.
- For details of each API Connect deployment profile, see: API Connect deployment profiles for OpenShift and Cloud Pak for Integration

- **API Connect deployment profiles for OpenShift and Cloud Pak for Integration**
  Details of the API Connect deployment profiles that are provided for OpenShift and Cloud Pak for Integration.
- **Component profiles, CPU limits, memory limits, and licensing**
  Available component profiles, their CPU and memory limits, and how the CPU-based licensing works.

API Connect deployment profiles for OpenShift and Cloud Pak for Integration

Details of the API Connect deployment profiles that are provided for OpenShift and Cloud Pak for Integration.

API Connect provides various profiles to suit your capacity and availability requirements.

The API Connect deployment profiles provide one Management component instance, and one instance of the Portal, Analytics, and Gateway components.

On Cloud Pak for Integration:

- The available deployment profiles are presented as tiles in the Platform UI.
- The Portal, Analytics, and Gateway components are automatically registered with the Management component during installation.

n1xc7.m48: Small - One replica

Best suited for light, noncritical workloads such as small team development and testing. Deploys 1 replica of the Management, Developer Portal, Analytics, and Gateway components with a minimal footprint.

Table 1. n1xc7.m48

Component name	Component profile
Management	n1xc2.m16
Portal	n1xc2.m8
Analytics	n1xc2.m16
Gateway	n1xc1.m8

n1xc16.m72: Medium - One replica

Ideal for medium workloads such as organizational development, testing, and production when in alignment with business availability requirements. Deploys 1 replica of the Management, Developer Portal, Analytics, and Gateway components.

Table 2. n1xc16.m72

Component name	Component profile
APIM	n1xc4.m16
Portal	n1xc4.m16
Analytics	n1xc4.m32
Gateway	n1xc4.m8

n3xc16.m48: Large - Three replicas

Ideal for heavy workloads when business requires redundancy for greater availability & capacity. Deploys 3 replicas of the Management, Developer Portal, Analytics, and Gateway components.

Table 3. n3xc16.m48

Component name	Component profile
APIM	n3xc4.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc4.m8

n1xc12.m64: Medium - One replica - Remote Gateway

Ideal for medium workloads such as organizational development, testing, and production when in alignment with business availability requirements. Must be configured for use with existing or independently deployed Gateways. Deploys 1 replica of the Management, Developer Portal, and Analytics components.

Table 4. n1xc12.m64

Component name	Component profile
APIM	n1xc4.m16
Portal	n1xc4.m16
Analytics	n1xc4.m32

n3xc12.m40: Large - Three replicas - Remote Gateway

Ideal for heavy workloads when business requires redundancy for greater availability & capacity. Must be configured for use with existing or independently deployed Gateways. Deploys 3 replicas the Management, Developer Portal, and Analytics components.

Table 5. n3xc12.m40

Component name	Component profile
APIM	n3xc4.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16

Deprecated profiles

The API Connect deployment profiles presented here existed in previous releases, but are now in a deprecated state and should not be used for new installations.

Table 6. n3xc11.m48

Component name	Component profile
APIM	n3xc2.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc1.m8

Table 7. n3xc14.m48

Component name	Component profile
APIM	n3xc2.m16
Portal	n3xc4.m8
Analytics	n3xc4.m16
Gateway	n3xc4.m8

Table 8. n1xc10.m48

Component name	Component profile
APIM	n1xc2.m16
Portal	n1xc2.m8
Analytics	n1xc2.m16
Gateway	n1xc4.m8

Component profiles, CPU limits, memory limits, and licensing

Available component profiles, their CPU and memory limits, and how the CPU-based licensing works.

API Connect is licensed based on the CPU usage of your deployment. Each component profile sets specific CPU and memory limits on each of the containers that comprise API Connect.

API Connect operator

The API Connect operator container uses the same CPU and memory requests and limits for all profiles, and is not a container that requires license entitlement.

Table 1. Operator container CPU and memory requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
ibm-apiconnect	ibm-apiconnect	75	1000	128	1Gi		1	No

The CPU and memory usage tables for each API Connect component profile are shown in the following pages.

- **Management component deployment profile limits**
  The CPU, memory limits, and PVC space that are required for each Management profile.
- **Portal component deployment profile limits**
  The CPU and memory limits that are defined for each Portal deployment profile.
- **Analytics component deployment profile limits**
  The CPU and memory limits that are defined for each Analytics deployment profile.
- **DataPower gateway deployment profile limits**
  The CPU and memory limits that are defined for each Gateway deployment profile.

Management component deployment profile limits

The CPU, memory limits, and PVC space that are required for each Management profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: `<total licensed CPU limit> = <total CPU limits of all licensed containers> * <number of replicas>`. For example, for the `n3xc2.m16` profile:

- The three licensed containers are `apim`, `postgres`, and `taskmanager`.
- These three containers have CPU limits of 625, 750, and 625.
- This profile is a three replica (n3) profile.

The total CPU limit of the licensed containers then is: (625 + 750 + 625) * 3 = 6000

All Management component deployment profiles have job pods that complete tasks during startup, upgrade, backup, and restore operations. The job pods do not have any licensed containers, and their CPU/memory requests and limits are the same for all profiles.

Table 1. Management job pods CPU and memory requirements

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
management-up-apim-schema	management-up-apim-schema-0-to-<nnn>-<xx>-<yy>	apim-schema	50 m	500 m	128 Mi	256 Mi
management-up-apim-data-populate	management-up-apim-data-populate-0-to-<nnn>-<xx>-<yy>	apim-data-populate	50 m	600 m	256 Mi	4096 Mi
management-up-lur-schema	management-up-lur-schema-0-to-<nnn>-<xx>-<yy>	lur-schema	50 m	500 m	128 Mi	256 Mi
management-up-lur-data-populate	management-up-lur-data-populate-0-to-<nnn>-<xx>-<yy>	lur-data-populate	50 m	500 m	128 Mi	256 Mi
management-apim-restore-<xxx>	management-apim-restore-<xxx>-<yyy>	apim-restore	50 m	500 m	128 Mi	256 Mi
management-up-compliance-schema	management-up-compliance-schema-0-to-<nnn>-<xx>-<yy>	compliance-schema	50 m	500 m	128 Mi	256 Mi
management-up-compliance-data-populate	management-up-compliance-data-populate-0-to-<nnn>-<xx>-<yy>	compliance-data-populate	50 m	500 m	128 Mi	256 Mi

The following tables show the pod resource and license requirement for each Management component deployment profile. A separate table shows the requirements for optional features.
Important: The tables for the three replica profiles show the usage for a single replica in the three replica deployment. Except for containers that are indicated to be one replica, multiply by three to get the total usage.

n1xc2.m16

Table 2. n1xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40 m	800 m	128 Mi	512 Mi		1	
apim	apim	200 m	625 m	128 Mi	10 Gi		1	Yes
client-downloads-server	client-downloads	20 m	100 m	8 Mi	100 Mi		1	
juhu	juhu	30 m	800 m	192 Mi	512 Mi		1	
ldap	ldap	20 m	300 m	64 Mi	128 Mi		1	
lur	lur	20 m	500 m	128 Mi	256 Mi		1	
natscluster	natscluster	40 m	300 m	16 Mi	64 Mi		1	
portal-proxy	portal-proxy	8 m	100 m	128 Mi	256 Mi		1	
management-*<random string>*-db-1	postgres	600 m	750 m	2 Gi	8 Gi	150 Gi	1	Yes
taskmanager	taskmanager	120 m	625 m	512 Mi	10 Gi		1	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		1	
	ui	20 m	500 m	64 Mi	256 Mi		1	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		1	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 3. n1xc2.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	20 m	500 m	128 Mi	384 Mi		1	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		1	
ATM (API testing)	hub	hub	200 m	750 m	512 Mi	1 Gi		1	Yes
	turnstile	turnstile	7 m	50 m	48 Mi	96 Mi		1	Yes
AI Test	openapi-analyzer	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	
Autotest assist	autotest-service	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	20 m	300 m	64 Mi	256 Mi		1	Yes
2DCDR	postgres-remote	tunnel	30 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	40 m	500 m	128 Mi	256 Mi		1	

n1xc3.m16

Table 4. n1xc3.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40 m	800 m	128 Mi	512 Mi		1	
apim	apim	200 m	900 m	128 Mi	10 Gi		1	Yes
client-downloads-server	client-downloads	20 m	100 m	8 Mi	100 Mi		1	
juhu	juhu	30 m	800 m	192 Mi	512 Mi		1	
ldap	ldap	20 m	300 m	64 Mi	128 Mi		1	
lur	lur	20 m	500 m	128 Mi	256 Mi		1	
natscluster	natscluster	40 m	300 m	16 Mi	64 Mi		1	
portal-proxy	portal-proxy	8 m	100 m	128 Mi	256 Mi		1	
management-*<random string>*-db-1	postgres	600 m	1200 m	2 Gi	8 Gi	150 Gi	1	Yes
taskmanager	taskmanager	120 m	900 m	512 Mi	10 Gi		1	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		1	
	ui	20 m	500 m	64 Mi	256 Mi		1	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		1	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 5. n1xc3.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	20 m	500 m	128 Mi	384 Mi		1	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		1	
ATM (API testing)	hub	hub	200 m	750 m	512 Mi	1 Gi		1	Yes
	turnstile	turnstile	7 m	50 m	48 Mi	96 Mi		1	Yes
AI TestGen	openapi-analyzer	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	
Autotest assist	autotest-service	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	20 m	300 m	64 Mi	256 Mi		1	Yes
2DCDR	postgres-remote	tunnel	30 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	40 m	500 m	128 Mi	256 Mi		1	

n1xc4.m16

Table 6. n1xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	100 m	800 m	128 Mi	512 Mi		1	
apim	apim	500 m	1250 m	128 Mi	10 Gi		1	Yes
client-downloads-server	client-downloads	50 m	100 m	8 Mi	100 Mi		1	
juhu	juhu	75 m	800 m	192 Mi	512 Mi		1	
ldap	ldap	50 m	300 m	64 Mi	128 Mi		1	
lur	lur	50 m	500 m	128 Mi	256 Mi		1	
natscluster	natscluster	100 m	300 m	16 Mi	64 Mi		1	
portal-proxy	portal-proxy	20 m	100 m	128 Mi	256 Mi		1	
management-*<random string>*-db-1	postgres	600 m	1500 m	2 Gi	8 Gi	150 Gi	1	Yes
taskmanager	taskmanager	300 m	1250 m	512 Mi	10 Gi		1	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		1	
	ui	50 m	500 m	64 Mi	256 Mi		1	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		1	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 7. n1xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	50 m	500 m	128 Mi	384 Mi		1	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		1	
ATM (API testing)	hub	hub	500 m	750 m	512 Mi	1 Gi		1	Yes
	turnstile	turnstile	16 m	50 m	48 Mi	96 Mi		1	Yes
AITestGen	openapi-analyzer	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	
Autotest assist	autotest-service	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	50 m	300 m	64 Mi	256 Mi		1	Yes
2DCDR	postgres-remote	tunnel	75 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	100 m	500 m	128 Mi	256 Mi		1	

n3xc2.m16

Table 8. n3xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40 m	800 m	128 Mi	512 Mi		3	
apim	apim	200 m	625 m	128 Mi	10 Gi		3	Yes
client-downloads-server	client-downloads	20 m	100 m	8 Mi	100 Mi		3	
juhu	juhu	30 m	800 m	192 Mi	512 Mi		3	
ldap	ldap	20 m	300 m	64 Mi	128 Mi		3	
lur	lur	20 m	500 m	128 Mi	256 Mi		3	
natscluster	natscluster	40 m	300 m	16 Mi	64 Mi		3	
portal-proxy	portal-proxy	8 m	100 m	128 Mi	256 Mi		3	
management-<*random string*>-db-n	postgres	600 m	750 m	2 Gi	8 Gi	230 Gi	3	Yes
taskmanager	taskmanager	120 m	625 m	512 Mi	10 Gi		3	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		3	
	ui	20 m	500 m	64 Mi	256 Mi		3	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		3	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 9. n3xc2.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	20 m	625 m	128 Mi	512 Mi		3	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		3	
ATM (API testing)	hub (one replica)	hub	200 m	750 m	1 Gi	2 Gi		1	Yes
	turnstile (one replica)	turnstile	40 m	250 m	128 Mi	256 Mi		1	Yes
AI TestGen	openapi-analyzer (one replica)	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
Autotest assist	autotest-service(one replica)	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	20 m	300 m	64 Mi	256 Mi		3	Yes
2DCDR	postgres-remote	tunnel	30 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	40 m	500 m	128 Mi	256 Mi		1	

n3xc3.m16

Table 10. n3xc3.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	40 m	800 m	128 Mi	512 Mi		3	
apim	apim	200 m	900 m	128 Mi	10 Gi		3	Yes
client-downloads-server	client-downloads	20 m	100 m	8 Mi	100 Mi		3	
juhu	juhu	30 m	800 m	192 Mi	512 Mi		3	
ldap	ldap	20 m	300 m	64 Mi	128 Mi		3	
lur	lur	20 m	500 m	128 Mi	256 Mi		3	
natscluster	natscluster	40 m	300 m	16 Mi	64 Mi		3	
portal-proxy	portal-proxy	8 m	100 m	128 Mi	256 Mi		3	
management-*<random string>*-db-n	postgres	600 m	1200 m	2 Gi	8 Gi	230 Gi	3	Yes
taskmanager	taskmanager	120 m	900 m	512 Mi	10 Gi		3	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		3	
	ui	20 m	500 m	64 Mi	256 Mi		3	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		3	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 11. n3xc3.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	20 m	625 m	128 Mi	512 Mi		3	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		3	
ATM (API testing)	hub (one replica)	hub	200 m	750 m	1 Gi	2 Gi		1	Yes
	turnstile (one replica)	turnstile	40 m	250 m	128 Mi	256 Mi		1	Yes
AI TestGen	openapi-analyzer (one replica)	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
Autotest assist	autotest-service (one replica)	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	20 m	300 m	64 Mi	256 Mi		3	Yes
2DCDR	postgres-remote	tunnel	30 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	40 m	500 m	128 Mi	256 Mi		1	

n3xc4.m16

Table 12. n3xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
analytics-proxy	analytics-proxy	100 m	800 m	128 Mi	512 Mi		3	
apim	apim	500 m	1250 m	128 Mi	10 Gi		3	Yes
client-downloads-server	client-downloads	50 m	100 m	8 Mi	100 Mi		3	
juhu	juhu	75 m	800 m	192 Mi	512 Mi		3	
ldap	ldap	50 m	300 m	64 Mi	128 Mi		3	
lur	lur	50 m	500 m	128 Mi	256 Mi		3	
natscluster	natscluster	100 m	300 m	16 Mi	64 Mi		3	
portal-proxy	portal-proxy	20 m	100 m	128 Mi	256 Mi		3	
management-*<random string>*-db-n	postgres	600 m	1500 m	2 Gi	8 Gi	230 Gi	3	Yes
taskmanager	taskmanager	300 m	1250 m	512 Mi	10 Gi		3	Yes
ui	ui-authenticaton	50 m	50 m	64 Mi	64 Mi		3	
	ui	50 m	500 m	64 Mi	256 Mi		3	
websocket-proxy	websocket-proxy	20 m	100 m	128 Mi	256 Mi		3	
edb-operator	manager	100 m	100 m	100 Mi	200 Mi			

Table 13. n3xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
API governance	management-compliance-service	compliance-service	50 m	625 m	128 Mi	512 Mi		3	
	management-compliance-ui	compliance-ui	20 m	100 m	64 Mi	128 Mi		3	
ATM (API testing)	hub (one replica)	hub	500 m	750 m	1 Gi	2 Gi		1	Yes
	turnstile (one replica)	turnstile	100 m	250 m	128 Mi	256 Mi		1	Yes
AI TestGen	openapi-analyzer (one replica)	openapi-analyzer	500 m	500 m	2 Gi	2 Gi		1	Yes
	api-testgen-data	api-testgen-data	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-coordinator	api-testgen-coordinator	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-discovery	api-testgen-discovery	300 m	300 m	384 Mi	384 Mi		1	
	api-testgen-generator	api-testgen-generator	300 m	300 m	384 Mi	384 Mi		1	

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
Autotest assist	autotest-service (one replica)	autotest-service	200 m	200 m	256 Mi	256 Mi		1	Yes
Billing	billing	billing	50 m	300 m	64 Mi	256 Mi		3	Yes
2DCDR	postgres-remote	tunnel	75 m	150 m	512 Mi	512 Mi		1	
	tunnel	server	100 m	500 m	128 Mi	256 Mi		1	

Portal component deployment profile limits

The CPU and memory limits that are defined for each Portal deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: **`<total licensed CPU limit> = <total CPU limits of all licensed containers> * <number of replicas>`**

All Portal component deployment profiles have a **`sync-backup`** job pod that completes tasks that are related to scheduled portal backups. The **`sync-backup`** job pod does not have any licensed containers, and its CPU/memory requests and limits are the same for all profiles.

Table 1. Portal job pods resource and licensing requirements

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
<portal-instance-name>-sync-backup	*<portal-instance-name>*-sync-backup-*<jobuuid>*	sync-ptl-backup-schedule	1m	1m	100Mi	100Mi

n1xc2.m8

Table 2. n1xc2.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	200m	2000m	3Gi	8Gi	6Gi	1	
db	dbproxy	100m	1000m	128Mi	512Mi	15Gi	1	
www	web	150m	2000m	1Gi	5Gi	8Gi	1	Yes
www	admin	200m	2000m	1Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	1	

Table 3. n1xc2.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	150m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	150m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the **`db-remote`** and **`www-remote`** pods have three replicas to match the remote site.

n1xc4.m16

Table 4. n1xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	500m	3000m	6Gi	8Gi	6Gi	1	
	dbproxy	450m	1000m	256Mi	1Gi	15Gi	1	
www	web	600m	4000m	3Gi	8Gi	8Gi	1	Yes
	admin	800m	4000m	3Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	1	

Table 5. n1xc4.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the `db-remote` and `www-remote` pods have three replicas to match the remote site.

n1xc8.m16

Table 6. n1xc8.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	1300m	5000m	6Gi	8Gi	6Gi	1	
	dbproxy	1000m	1000m	256Mi	1Gi	15Gi	1	
www	web	1300m	8000m	3Gi	8Gi	8Gi	1	Yes
	admin	2300m	4000m	3Gi	8Gi	6Gi + 15Gi	1	
nginx	nginx	100m	500m	512Mi	512Mi	1Gi	1	

Table 7. n1xc8.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	1	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	1	

Note: For 2DCDR deployments, if the remote site is deployed with a three replica profile, the `db-remote` and `www-remote` pods have three replicas to match the remote site.

n3xc3.m8

Table 8. n3xc3.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	400m	2000m	3Gi	8Gi	12Gi	3	
	dbproxy	150m	1000m	128Mi	512Mi	30Gi	3	
www	web	450m	3000m	1Gi	5Gi	12Gi	3	Yes
	admin	650m	3000m	1Gi	8Gi	12Gi + 30Gi	3	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	3	

Table 9. n3xc3.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the `db-remote` and `www-remote` pods have one replica, to match the remote site.

n3xc4.m8

Table 10. n3xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	500m	3000m	3Gi	8Gi	12Gi	3	
	dbproxy	450m	1000m	128Mi	512Mi	30Gi	3	
www	web	600m	4000m	1Gi	5Gi	12Gi	3	Yes
	admin	800m	4000m	1Gi	8Gi	12Gi + 30Gi	3	
nginx	nginx	20m	500m	512Mi	512Mi	1Gi	3	

Table 11. n3xc4.m8 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the `db-remote` and `www-remote` pods have one replica, to match the remote site.

n3xc8.m16

Table 12. n3xc8.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
db	db	1300m	5000m	6Gi	8Gi	6Gi	3	
	dbproxy	1000m	1000m	256Mi	1Gi	15Gi	3	
www	web	1300m	8000m	3Gi	8Gi	8Gi	3	Yes
	admin	2300m	4000m	3Gi	8Gi	6Gi + 15Gi	3	
nginx	nginx	100m	500m	512Mi	512Mi	1Gi	3	

Table 13. n3xc8.m16 Containers for optional features, resource and licensing requirements

Feature	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
2DCDR	db-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	www-remote	tunnel	75m	500m	160Mi	256Mi	1Gi	3	
	tunnel	server	100m	500m	256Mi	512Mi	1Gi	3	

Note: For 2DCDR deployments, if the remote site is deployed with a one replica profile, the **db-remote** and **www-remote** pods have one replica, to match the remote site.

Analytics component deployment profile limits

The CPU and memory limits that are defined for each Analytics deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: **<total licensed CPU limit> = <total CPU limits of all licensed containers> * <number of replicas>**

All Analytics component deployment profiles have an **osinit** job pod that is run on startup and completes tasks that are related to initialization. The **osinit** job pod does not have any licensed containers, and its CPU/memory requests and limits are the same for all profiles.

You can choose different storage types with the three replica profiles, and so the pods deployed differ depending on the storage option chosen: Analytics storage type. If you choose storage type **shared**, then your analytics deployment has 1 storage pod, called **storage**. If you choose storage type **dedicated**, then your analytics deployment has 2 storage pods, called **storage** and **storage-os-master**. The tables for the three replica profiles indicate in parentheses after the pod name which storage type the pod applies to, for example "storage (shared)".

Table 1. Analytics job pods

Job Name	Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits
<analytics-instance-name>-osinit	*<analytics-instance-name>*-osinit-*<podid>*	osinit	8 m	100 m	128 Mi	256 Mi
<analytics-instance-name>-oscron	*<analytics-instance-name>*-oscron-*<jobid>*-*<podid>*	oscron	8 m	100 m	128 Mi	256 Mi

The PVC sizes that are specified in the tables are the defaults. Review the planning and installation documentation to decide on the sizes to allocate for your deployment: Planning your analytics deployment.

n1xc2.m16

Table 2. n1xc2.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	128 Mi	256 Mi		1	
ingestion	ingestion	275 m	1200 m	8 Gi	8 Gi	50 Gi	1	Yes
storage	storage	225 m	800 m	6 Gi	6 Gi	200 Gi	1	Yes
director	director	50 m	200 m	128 Mi	512 Mi		1	

n1xc4.m32

Table 3. n1xc4.m32 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		1	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	50 Gi	1	Yes
storage	storage	1200 m	1800 m	22 Gi	22 Gi	200 Gi	1	Yes
director	director	50 m	200 m	256 Mi	512 Mi		1	

n1xc6.m48

Table 4. n1xc6.m48 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		1	
ingestion	ingestion	2000 m	3200 m	8 Gi	8 Gi	50 Gi	1	Yes

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
storage	storage	1800 m	2800 m	37 Gi	37 Gi	200 Gi	1	Yes
director	director	50 m	200 m	256 Mi	512 Mi		1	

n3xc4.m16

Table 5. n3xc4.m16 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	128 Mi	256 Mi		3	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	50 Gi	3	Yes
director	director	50 m	200 m	128 Mi	256 Mi		3	
storage (shared)	storage	1200 m	1800 m	6 Gi	6 Gi	200 Gi	3	Yes
storage (dedicated)	storage	800 m	1000 m	4 Gi	4 Gi	200 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	800 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc4.m32

Table 6. n3xc4.m32 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	1300 m	2200 m	8 Gi	8 Gi	50 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	1200 m	1800 m	22 Gi	22 Gi	200 Gi	3	Yes
storage (dedicated)	storage	800 m	1000 m	20 Gi	20 Gi	200 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	800 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc6.m48

Table 7. n3xc6.m48 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	2000 m	3200 m	8 Gi	8 Gi	50 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	1800 m	2800 m	37 Gi	37 Gi	200 Gi	3	Yes
storage (dedicated)	storage	1400 m	1800 m	35 Gi	35 Gi	200 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master	400 m	1000 m	2 Gi	2 Gi	5 Gi	3	Yes

n3xc8.m64

Table 8. n3xc8.m64 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
mtls-gw	mtls-gw	100 m	1000 m	256 Mi	512 Mi		3	
ingestion	ingestion	3000 m	4200 m	8 Gi	8 Gi	50 Gi	3	Yes
director	director	50 m	200 m	256 Mi	512 Mi		3	
storage (shared)	storage	2800 m	3800 m	53 Gi	53 Gi	200 Gi	3	Yes
storage (dedicated)	storage	2400 m	2800 m	49 Gi	49 Gi	200 Gi	3	Yes
storage-os-master (dedicated)	storage-os-master (dedicated)	400 m	1000 m	4 Gi	4 Gi	5 Gi	3	Yes

DataPower gateway deployment profile limits

The CPU and memory limits that are defined for each Gateway deployment profile.

The tables in this topic also show which containers require license entitlement. The sum of the CPU limits of the licensed containers, multiplied by the number of replicas determine what you are charged. The formula is: `<total licensed CPU limit> = <total CPU limits of all licensed containers> * <number of replicas>`.

Note: All profiles show PVC size as 30Gi, this is only required if the Token Management Service (TMS) is enabled. If TMS is not enabled, there is no PVC requirement.

n1xc1.m8

Table 1. n1xc1.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	1000m	1000m	8Gi	8Gi	30Gi	1	Yes

n1xc4.m8

Table 2. n1xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	4000m	4000m	8Gi	8Gi	30Gi	1	Yes

n3xc1.m8

Table 3. n3xc1.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	1000m	1000m	8Gi	8Gi	30Gi	3	Yes

n3xc4.m8

Table 4. n3xc4.m8 profile resource and licensing requirements

Pod Name	Container Name	CPU Requests	CPU limits	Memory Requests	Memory Limits	PVC Size	Number of replicas	License entitlement required
gateway	datapower	4000m	4000m	8Gi	8Gi	30Gi	3	Yes

Multiple data center deployment strategies

Strategies for deploying API Connect across multiple data centers.

If you want to deploy API Connect across multiple data centers, then several options are available and are detailed in this topic.

Note:
You can install the subsystems that make up an API Connect deployment in different data centers, but you should not install a single API Connect subsystem across data centers. For example, if you are deploying a three replica analytics subsystem, then all analytics subsystem replicas should run in the same data center.

There is an exception for the gateway service: If TMS is required and an external OAuth provider cannot be used, then the gateways can be located in different data centers if network latency between data centers is less than 30 ms.

Remote gateway deployment

To provide high availability for your published APIs, you can deploy one or more gateway services in different data centers, and register them with the management subsystem in your main data center. If your main data center has an outage, then your remote gateways continue to process calls to your published APIs.

Remote gateway deployment is the simplest way to take advantage of multiple data centers. The main disadvantage is that you cannot update any published APIs during the main data center outage (because your management subsystem is in the main data center).

To install API Connect with remote gateways, follow the multi-namespace installation procedures for your platform:

- OpenShift: Installing API Connect subsystems in different namespaces or environments on OpenShift.
- Kubernetes: Deploying operators in a multi-namespace API Connect cluster.

Cold-standby deployment

In a cold-standby deployment, API Connect is installed and running in your main data center. In your cold-standby data center, API Connect is ready to be installed and has access to the API Connect backup files from your main data center.

If your main data center has an outage, then you install API Connect in your cold-standby data center, and restore your most recent subsystem database backups to your cold-standby deployment.

Requirements for cold-standby configuration:

- The API Connect deployment in the cold-standby data center must use the same deployment profile as the main data center.
- The API Connect network endpoints that you define in your main data center must be valid in your cold-standby data center.
  For example, if the network endpoint for your Cloud Manager UI is `cloud-admin.apicinstall.example.com`, this endpoint must route correctly to the management subsystem you install in your cold-standby data center.

  Every network endpoint that you configure during API Connect installation and configuration in your main data center must be valid in your cold-standby data center.

- Ensure that subsystem backups are regularly copied from your main data center to your cold-standby. Backups must include both your infrastructure configuration (CR and secret YAML files), and subsystem database backups. See Backing up, restoring, and disaster recovery.

The steps for failing over to your cold-standby data center are:

1. Install API Connect in your cold-standby data center, by using the infrastructure configuration backup YAML files from your main data center.
2. Restore your API Connect subsystem database backups.
3. Update network routing so that user traffic is directed to your cold-standby data center.

Steps 1 and 2 are essentially the same as the disaster recovery steps to restore API Connect, see Backing up, restoring, and disaster recovery.

The advantage of the cold-standby deployment over the remote gateway deployment is that your full API Connect deployment is available in the cold-standby site. Management and portal subsystem users can update APIs and view analytics data while your main site is unavailable.

Warm-standby deployment

In the warm-standby deployment, the management and portal subsystems continually replicate their databases to a warm-standby API Connect deployment in a separate data center.

The advantage of the warm-standby deployment over cold-standby is that the failover process is faster, and the management and portal databases in the warm-standby site are fully up to date. With the cold-standby strategy, your management and portal databases are only as recent as the last backup you took from your main site.

The disadvantage of the warm-standby deployment is that it adds extra complexity to the API Connect installation process, and all ongoing maintenance procedures such as upgrade. Use the warm-standby deployment strategy only when the following are essential requirements:

- Minimal management and portal subsystem downtime.
- No management and portal data loss, when restoration of data from the last backup that was taken of your active site is not sufficient.

Requirements for warm-standby configuration:

- The API Connect deployment in the warm-standby data center must use the same deployment profile as the main data center.
- The API Connect network endpoints that you define in your main data center must be valid in your warm-standby data center.
  For example, if the network endpoint for your Cloud Manager UI is `cloud-admin.apicinstall.example.com`, this endpoint must route correctly to the management subsystem you install in your warm-standby data center.

  Every network endpoint that you configure during API Connect installation and configuration in your main data center must be valid in your warm-standby data center.

The warm-standby deployment configuration is referred to in this documentation as two data center disaster recovery (2DCDR). For more information about 2DCDR, see Two data center warm-standby deployment on Kubernetes and OpenShift.

- **Two data center warm-standby deployment on Kubernetes and OpenShift**
  An overview of the two data center disaster recovery deployment strategy with a warm-standby data center.

Two data center warm-standby deployment on Kubernetes and OpenShift

An overview of the two data center disaster recovery deployment strategy with a warm-standby data center.

The two data center disaster recovery warm-standby deployment configuration (referred to as 2DCDR) provides continuous replication of the management and portal subsystem databases to a warm-standby deployment in another data center.
Important: The 2DCDR deployment adds complexity to the API Connect installation procedures, and all subsequent maintenance procedures such as backup, restore, and upgrade. Before you decide to use the 2DCDR deployment strategy, review the Multiple data center deployment strategies to see whether any of the alternatives are better suited to your requirements.

Key points of the two data center disaster recovery (DR) solution

- Two data center DR is an active/warm-standby deployment for the API Manager and Developer Portal services, and must use manual failover.
- Two DataPower® Gateway subsystems must be deployed to provide high availability for the gateway service. However, this scenario doesn't provide high availability for the analytics service.
- If high availability is required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.
- Data consistency is prioritized over data availability.
- Each data center must be setup within its own Kubernetes or OpenShift cluster.
- Latency between the two data centers must be less than 80 ms.
- Port 443 must be open between data centers for API Connect database replication.
- Replication of the API Manager is asynchronous, so it is possible that the most recent updates do not transfer to the warm-standby data center if there is an active data center failure.
- Replication of the Developer Portal is synchronous, and therefore the latency is limited to 80 ms or less.
- The API Manager and the Developer Portal services in the two data centers must use the same deployment profile.
- The deployment in each data center is effectively an instance of the same API Connect deployment - therefore, the endpoints, certificates, and Kubernetes secrets must all be the same.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a two data center disaster recovery configuration.

Deployment architecture

A two data center deployment model is optimized for data consistency ahead of data availability, and must use manual failover when a fault occurs. For high availability of Management and Portal subsystems ensure that you use a three replica deployment profile. For more information on deployment profiles, see: Deployment and

component profiles.

For a single Kubernetes cluster to span multiple data centers the network latency must be no more than a few milliseconds, typically less than 10 ms. This low latency is often unachievable between geographically separated data centers. For this reason, the two data center disaster recovery solution requires that each data center is set up with its own Kubernetes cluster. The Management and Portal subsystem databases are continually replicated from the active datacenter to the warm-standby datacenter, this requires a network latency between the two data centers of less than 80 ms.

To achieve high availability for the DataPower Gateway, you must deploy two gateway subsystems. One subsystem in the active data center, and a separate subsystem in the warm-standby data center. Publish all Products and APIs to both gateway subsystems. The gateway subsystems are independent, and so are insulated if an issue occurs in one of them. A global dynamic router can then be used to route traffic to one gateway subsystem or the other. If high availability is also required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.

There can be multiple Developer Portal services in an API Connect deployment (although still only one Developer Portal site per Catalog).

The deployment in each data center is effectively an instance of the same API Connect deployment – as the database is replicated, all of the configuration is also shared. Therefore, the endpoints, certificates, and Kubernetes secrets all need to be the same.

Dynamic routing
: Note: A dynamic routing device, such as a load balancer, is required to route traffic to either data center. However, neither this device nor its configuration is part of the API Connect offering. Contact IBM Services if you require assistance with configuring a dynamic router.
The dynamic router configuration must handle the traffic between the subsystems and between the data centers. For example, the consumer API calls from the Developer Portal to API Manager must transfer through a dynamic router so that the Developer Portal can use a single endpoint regardless of which data center API Manager is active in. The same is needed for calls to the Platform and Admin APIs from the other subsystems, as well as for incoming UI traffic for the Developer Portal UI, Cloud Manager UI, and API Manager UI.

: The dynamic router must support SSL passthrough, so that it routes the Mutual TLS (mTLS) connections between API Manager and the Developer Portal, and between API Manager and the DataPower Gateway. The router should not do TLS termination, it should do layer 4 based routing by using SNI.
Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see Enable JWT security instead of mTLS.

Service status
: When a failure occurs, it is common practice to display an interstitial system outage web page. The dynamic router can be configured to display this web page when there is a failure, while the manual failover to the warm-standby data center is taking place.

Deployment-profiles
: Both one replica and three replica deployment profiles can be used with two data center DR, but they must be the same at each data center. For more information about the deployment-profiles, see Deployment and component profiles on Kubernetes, and Requirements for initial deployment on VMware.

For more information about a two data center DR deployment, see the following topics:

Kubernetes and OpenShift

- Installing a two data center deployment on Kubernetes
  - Converting a single data center to a two data center deployment on Kubernetes
- Maintaining a two data center deployment

OpenShift: Deciding to use individual CRs, a top-level CR, or Cloud Pak for Integration

You can install API Connect subsystems separately by using individual subsystem CRs, or you can deploy them all together with a top-level CR. You can also deploy API Connect as part of Cloud Pak for Integration. The differences between each installation type are described in this topic.

Before you review this topic, make sure that you understand the Key concepts: Custom Resources, Operators, and Operands of API Connect.

Deploying with individual subsystem CRs

Deploying API Connect using individual subsystem CRs means that you must install each subsystem separately, either with the OpenShift UI, or by manually applying YAML files for each subsystem. The portal, analytics, and gateway subsystems must then be manually registered with the management subsystem, as described in Defining your Cloud Manager topology.

Individual subsystem CRs: Key points

- Deploying with individual subsystem CRs allows more flexibility than the alternative top-level CR deployment, and can be simpler to manage.
  - Subsystems in the same API Connect cluster can be installed in different namespaces.
  - Updates to subsystem CRs are made directly in the subsystem CR. When the top-level CR is used, all configuration updates must be made in the top-level CR, which can require copying sections from the subsystem CR and pasting them into the top-level CR.
  - Custom certificates can be used for inter-subsystem communication.
  - The two data center disaster recovery deployment topology is simpler to manage and more flexible when individual subsystem CRs are used.
- Individual subsystem CR installation does not integrate as fully with Cloud Pak for Integration as the top-level CR installation. See *API management deployment* in the Cloud Pak for Integration documentation.
- The individual subsystem CR installation is preferable for deployments where you plan to have multiple portal and analytics services, or subsystems in different namespaces, and where you do not require the other capabilities of Cloud Pak for Integration.

Deploying with the top-level CR

Deploying with the top-level CR automates the deployment of each subsystem. To install, you select 'API Connect' in either the OpenShift console or the Cloud Pak for Integration Platform UI, enter some properties such as namespace, click 'Create', and all the subsystems that make up an API Connect instance are automatically installed. You still have individual CRs for each subsystem, but they are all managed by the top-level CR. The top-level CR is called **`APIConnectCluster`**.

Top-level CR: Key points to consider

- All API Connect subsystems are managed by the top-level CR. You edit the top-level CR YAML when you want to update your API Connect configuration.
- With the top-level CR, the subsystem CRs still exist but should not be edited directly. Any changes made to them get overwritten by the top-level CR.
- Installation with the top-level CR requires fewer steps, but subsequent management of subsystems and configuration of advanced features can be more complicated than it is with an individual subsystem deployment.
- The top-level CR deployment provides one of each subsystem service: management, portal, analytics, and gateway. If you want additional portal, analytics, or gateway services, do not use the top-level CR.
- For a quick proof-of-concept demonstration, the top-level CR is ideal due to the automated installation of subsystems.

Deploying with Cloud Pak for Integration

Cloud Pak for Integration includes API Connect as one of its many capabilities. For information on everything Cloud Pak for Integration provides, see https://www.ibm.com/docs/en/cloud-paks/cp-integration.

Key points of a Cloud Pak for Integration API Connect installation:

- Cloud Pak for Integration provides a UI, called the Platform UI, which can be used to install and manage your API Connect deployment.
- When installing API Connect with the Cloud Pak for Integration Platform UI, the top-level CR deployment is used. All the same key points of the top-level CR apply to Cloud Pak for Integration installations that use the top-level CR.
- API Connect automatically incorporates and attempts to use Cloud Pak features if installed in an environment where Cloud Pak for Integration is already installed. You can prevent Cloud Pak for Integration incorporating your API Connect installation by setting `metadata.annotations.apiconnect-operator/cp4i` to `false` in the top-level CR before installation.
- It is not possible to uninstall Cloud Pak for Integration and retain any API Connect instance that it is managing. Nor can a Cloud Pak for Integration API Connect instance be converted to a stand-alone instance. If you want to change to a non-Cloud Pak for Integration deployment, see Migrating from v10 to v10 on a different form factor.
- You can migrate an existing top-level CR installation to be incorporated into Cloud Pak for Integration. See see Migrating from OpenShift to Cloud Pak for Integration on the same cluster.
  Note: The reverse of this operation is not supported.
- In a Cloud Pak for Integration installation, if the Platform UI is detected and the top-level CR deployment is used, the API Connect operator runs a job that is called the `configurator`, which registers all API Connect subsystems with the management service after installation. This means that your API Connect installation is ready for you to immediately start configuring provider organizations.
- Cloud Pak for Integration does mean a higher computing resource requirement in your environment, when compared to a stand-alone API Connect installation.
- The planning requirements for API Connect such as available PVs and Firewall settings that are described in Planning your deployment, also apply when you install API Connect with Cloud Pak for Integration.

Which deployment type should you choose?

For long-term ease of maintenance and maximum flexibility, choose Deploying with individual subsystem CRs.

For integration with the App Connect, Asset Repo, Platform UI, and other Cloud Pak for Integration features, choose Deploying with Cloud Pak for Integration.

For quick installation of API Connect that is suitable for demonstrations and testing, choose Deploying with the top-level CR.

Network requirements for inter-subsystem communication

Plan how to secure the network communications between your API Connect subsystems.

API Connect provides various configuration options that are related to inter-subsystem communication.

Support load-balancer TLS termination by disabling mTLS and enabling JWT security

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.
Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on load-balancers.
The following communication flows can be secured with JWT:

- Management initiated communication to portal, analytics, and gateway subsystems.
- Analytics ingestion: The flow of API event records from the gateway to the analytics subsystem.
  Note: If the gateway is under a high transaction load and sending many API events to the analytics subsystem, the enablement of JWT might impact the performance of your analytics subsystem. It is recommended to run performance tests under your expected analytics load before you decide to use JWT for the analytics ingestion path.

With JWT enabled, the portal, gateway, and analytics subsystems verify the JSON Web Token (JWT) sent from the management subsystem when it initiates communication with them. The analytics subsystem verifies the JWT sent from the gateway with incoming API event data. The subsystems that receive the token contact a JSON Web Key Set (JWKS) URL to verify it. The JWKS URL is hosted by the management subsystem, in a subpath of the management subsystem's platform REST API.
For more information about this feature, see Enable JWT security instead of mTLS.

Use services instead of routes (OpenShift) or ingresses (Kubernetes)

Note: This feature is available on Kubernetes and OpenShift deployments where the API Connect subsystems are within the same cluster. It is not available on two data center disaster recovery deployments: Two data center warm-standby deployment on Kubernetes and OpenShift.
For inter-subsystem communication, API Connect uses routes on OpenShift and Ingresses on Kubernetes. Both routes and ingresses require the exposure of external HTTPS endpoints. On API Connect deployments where all subsystems are in the same cluster, you can use services instead of routes or ingresses for inter-subsystem communication. When API Connect is configured to use services, all inter-subsystem communications are kept within the cluster, providing greater security and reduced load on external network infrastructure such as load-balancers.

In the context of inter-subsystem communication, the use of routes or ingresses is referred to as **external** communication, and the use of services is referred to as **in-cluster** communication.

You select whether to use **external** or **in-cluster** communication when you register your portal, gateway, or analytics subsystems with the management subsystem.

For more information about this feature, see In-cluster service communication between subsystems.

Enabling mTLS on management to gateway communication

On Kubernetes and on OpenShift installations where the top-level CR is not used, mTLS can be enabled for management to gateway communication. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add **mtlsValidateClient** to the **spec** section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under **gatewayManagerEndpoint.hosts.secretName**. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under **gateway.client.secretName**.
Note: In releases previous to 10.0.5.3, this property is called **validateApimClient**.
Note: If you do not want to use mTLS, the you must enable JWT instead.
Note: On Cloud Pak for Integration and OpenShift where the top-level CR is used, mTLS enablement for management to gateway communication is not supported.

Certificates used to secure inter-subsystem communication

For information about the management of the certificates that are used in the TLS and mTLS communications between subsystems, see Key Concepts: Cert-manager, Issuers, and Secrets.

Enable JWT security instead of mTLS

When and how to use JWT security instead of mTLS.

Most of the network interactions between API Connect subsystems use mTLS. The communication flows in API Connect that are secured with mTLS are as follows:

- Management to portal. mTLS set by default.
- Management to analytics. mTLS set by default.
- Management to API gateway. TLS set by default, mTLS is enabled following Enabling mTLS on management to gateway communication.
- Management to v5 compatible gateway. TLS set by default, mTLS is enabled following Enabling mTLS on management to gateway communication.
- Management to Event gateway. mTLS set by default. Cloud Pak for Integration only.
- API gateway to analytics. mTLS set by default.
- V5 compatible gateway to analytics. mTLS is set and cannot be disabled.
- Event gateway to analytics. mTLS set and cannot be disabled. Cloud Pak for Integration only.

All other inter-subsystem communication uses TLS. For the communication routes that use mTLS, it is a requirement that all network infrastructure such as load-balancers that are located between API Connect subsystems implement TLS passthrough (and not TLS termination).

Use of JSON Web Token (JWT) security instead of mTLS allows for the deployment of API Connect in network infrastructures where TLS termination is enabled on load-balancers that are located between subsystems. JWT provides application layer security between API Connect subsystems. The typical use-case for this feature is when payload inspection of all network communications is required at network boundaries. The communication flows that support JWT security are:

- Management to portal.
- Management to analytics.
- Management to API gateway.
- Management to v5 compatible gateway.
- API gateway to analytics.
  Note: It is not possible to use JWT and disable mTLS on the V5 compatible gateway to analytics message flow.

The management subsystem always sends a JWT when it initiates communication with the other subsystems, but by default the subsystems do not verify this JWT. When JWT security is enabled, the subsystems verify the JWT by making a REST call to the JSON Web Key Set (JWKS) URL. The JWKS URL is hosted on the management subsystem's platform REST API.

To enable JWT verification on management initiated communication, update the subsystem CRs to specify the JWKS URL. To disable JWT verification, update the subsystem CRs and delete the value of the JWKS URL (leave it unset).

To enable JWT verification on gateway to analytics communication, enable the Use JWT switch when you register the gateway in the Cloud Manager UI.
Note: For OpenShift users: The example steps in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place. If you are using a top-level CR you must edit the **APIConnectCluster** CR (the top-level CR), instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the **APIConnectCluster** CR.

Disabling mTLS and enabling JSON Web Token (JWT) security

If you disable mTLS, you must enable JWT. It is not possible to configure API Connect with both mTLS and JWT disabled.

You disable mTLS on the portal, gateway (for management to gateway communication), and analytics subsystems by setting the **mtlsValidateClient** property to false in the Custom Resources (CRs) of each subsystem.
Note: By default, **mtlsValidateClient** is set to false on the gateway subsystem.
When **mtlsValidateClient** is set to false, the subsystem does not verify the client's TLS credentials, but it does check for a valid JWT configuration. For example, to disable mTLS and enable JWT for management to portal communication, edit the portal CR and set:

```
spec:
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the `jwksUrl` specified in the management subsystem CR `status.endpoints` section.
Using registration of the portal service in the Cloud Manager UI as an example, JWT works as follows:

1. Management subsystem establishes a TLS connection to the portal subsystem on the portal director admin endpoint, and sends a JWT to the portal.
2. Before the portal trusts the management subsystem, it first verifies the JWT it received. The JWT is verified by a call to a JSON Web Key Set (JWKS) URL.
3. The management subsystem receives the JWT verification request from the portal on its platform REST API, and responds that the JWT is valid.
4. The portal receives confirmation that the JWT is valid, and so it can trust the management subsystem and continue the registration process.

To disable JWT security for management initiated communication, either remove the `jwksUrl` property from the CR, or leave its value empty. If you disable JWT, you must enable mTLS by setting `mtlsValidateClient` to true.

To disable JWT security for gateway to analytics communication, disable the Use JWT switch for the registered gateway in the Topology page of the Cloud Manager UI.

Configuring the JWKS URL

The JWKS URL is defined as the platform API hostname with the following subpath appended: `api/cloud/oauth2/certs`. You can see what the JWKS URL is from the `status.endpoints` section of the management CR, for example:

```
status:
...
  endpoints:
...
    - name: jwksUrl
      secretName: api-endpoint
      type: API
      uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

Note: The JWT feature is intended for environments where the are load-balancers located between subsystems. If you are using the in-cluster communications feature Use services instead of routes (OpenShift) or ingresses (Kubernetes), then you cannot have load-balancers between subsystems, and so disabling mTLS and enabling JWT is not necessary. However, if you do want to use the JWT feature with in-cluster communications then the JWKS URL can be obtained following these steps:

1. Describe any API Connect pod:

   ```
   kubectl describe pod <apic-pod>
   ```

2. In the pod describe output, search for `APIC_PLATFORM_API_SVC_ENDPOINT`:

   ```
   Containers:
     apim:
       Environment:
         APIC_PLATFORM_API_SVC_ENDPOINT:  https://management-juhu.acme-v1005-020323.svc:2000/api
   ```

3. The JWKS URL is the value of `APIC_PLATFORM_API_SVC_ENDPOINT` with `/cloud/oauth2/certs` appended to it. If the APIC_PLATFORM_API_SVC_ENDPOINT is missing the `/api` at the end, add `api/cloud/oauth2/certs` instead. The path of the JWKS URL must always be `api/cloud/oauth2/certs`.

Key points and limitations of JWT security

- For API Connect deployments upgraded from a pre-v10.0.5.3 release, the behavior of each subsystem remains the same as it was before upgrade.
- Changing the JWKS URL on portal, and analytics subsystems causes pods to restart, and so a short subsystem outage. The gateway pod does not restart when the JWKS URL is changed.
- On new installs of Cloud Pak for Integration and top-level CR OpenShift deployments, the JWKS URL is set automatically for the portal, gateway, and analytics subsystems.
- You can have both mTLS and JWT enabled, although the analytics subsystem does not verify the JWT if mTLS is enabled.
- The v5c gateway does not send a JWT to the analytics subsystem. This flow is secured by mTLS, which cannot be disabled.
- If you enable JWT for gateway to analytics communication, the ingestion of analytics data becomes dependent on the availability of the management subsystem (so that the JWT can be verified). For more information about this limitation, see: JWT usage between gateway and analytics subsystems.
- Under high analytics ingestion loads, JWT enablement might impact the performance of the analytics subsystem. It is recommended to complete a performance test under your expected analytics load before you decide to use JWT for gateway to analytics communication.

JWT usage between gateway and analytics subsystems

Because the analytics subsystem makes a REST call to the management subsystems to verify the JWT it receives from the gateway, analytics ingestion depends on the management subsystem when JWT is enabled. The analytics subsystem caches the response from the JWKS URL so that it does not need to make a REST call to the management subsystem every time API event data is received from the gateway. But the JWT verification cache does expire after 24 hours, and a new REST call to the management subsystem is made to refresh the cache. The JWT dependency on the platform REST API means that an outage of the management subsystem does eventually prevent the analytics subsystem from accepting new analytics API event records.

In-cluster service communication between subsystems

Key points and limitations of `in-cluster` inter-subsystem communication.

- In-cluster communication is only possible between subsystems that are in the same cluster.
- In-cluster communication cannot be used in two data center disaster recovery deployments, Two data center warm-standby deployment on Kubernetes and OpenShift.
- If you are adding new subsystems to an upgraded deployment you can set the subsystems to use `in-cluster` communication, but you must use different certificates and secrets for the subsystem endpoints. The default certificate and secret names for the subsystem endpoints are:
  - Analytics: `ai-endpoint`.

- Portal: `portal-admin`.
- Gateway: `gwv6-manager-endpoint` or `gw-gateway-manager`

Do not use these same certificate and secret names if your additional subsystems are in the same namespace.

- If you customize any TLS certificates used for inter-subsystem communication, then to use `in-cluster` communication the TLS certificates must include the service hostname in the DNS section of the SAN, for example:

```
X509v3 Subject Alternative Name: critical
                DNS: ptladmin.mydomain.com, DNS: portal.apic.svc, DNS: portal.apic.svc.cluster.local
```

- On Cloud Pak for Integration, all subsystems are registered automatically during deployment with `external` communication specified.

Kubernetes ingress controller prerequisites

Describes the prerequisite settings for the ingress controller for a Kubernetes runtime environment.

Before you begin

Note:

- This article does not apply to the IBM® DataPower® Gateway component that is a part of the IBM API Connect solution.
- This article applies to third-party software that IBM does not control. As such, the software may change and this information may become outdated.

These instructions assume you have a working Kubernetes environment and understand how to manage Kubernetes. Kubernetes is a platform for automated deployment, scaling, and operation of application containers across clusters of hosts, providing container-centric infrastructure. For more information, see https://kubernetes.io.

Kubernetes/ingress-nginx ingress controller `ingress-config.yml` settings

A Kubernetes deployment for IBM API Connect requires the **kubernetes/ingress-nginx** ingress controller implementation (see https://github.com/kubernetes/ingress-nginx) with SSL passthrough enabled.

API Connect v10 does not require Helm, so it is recommended to use Helm3 for the installation of the ingress controller. Follow these steps:

1. Create a file `ingress-config.yaml` where the following values are required:
   Specify at least one SSL protocol; separate multiple protocols with a comma as shown in the example.

```
controller:
  watchIngressWithoutClass: true
  admissionWebhooks:
    enabled: false
  config:
    ssl-protocols: "TLSv1.2 TLSv1.3"
  extraArgs:
    annotations-prefix: ingress.kubernetes.io
    enable-ssl-passthrough: true
```

   You may use the following sample ingress-config.yml file to configure the ingress controller:

```
controller:
  watchIngressWithoutClass: true
  admissionWebhooks:
    enabled: false
  config:
    hsts-max-age: "31536000"
    keepalive: "32"
    log-format: '{ "@timestamp": "$time_iso8601", "@version": "1", "clientip": "$remote_addr",
      "tag": "ingress", "remote_user": "$remote_user", "bytes": $bytes_sent, "duration":
      $request_time, "status": $status, "request": "$request_uri", "urlpath": "$uri",
      "urlquery": "$args", "method": "$request_method", "referer": "$http_referer",
      "useragent": "$http_user_agent", "software": "nginx", "version": "$nginx_version",
      "host": "$host", "upstream": "$upstream_addr", "upstream-status": "$upstream_status"
      }'
    main-snippets: load_module "modules/ngx_stream_module.so"
    proxy-body-size: "0"
    proxy-buffering: "off"
    server-name-hash-bucket-size: "128"
    server-name-hash-max-size: "1024"
    server-tokens: "False"
    ssl-ciphers: HIGH:!aNULL:!MD5
    ssl-prefer-server-ciphers: "True"
    ssl-protocols: "TLSv1.2 TLSv1.3"
    use-http2: "true"
    worker-connections: "10240"
    worker-cpu-affinity: auto
    worker-processes: "1"
    worker-rlimit-nofile: "65536"
    worker-shutdown-timeout: 5m
  daemonset:
    useHostPort: false
  extraArgs:
    annotations-prefix: ingress.kubernetes.io
    enable-ssl-passthrough: true
  hostNetwork: true
  kind: DaemonSet
  name: controller
rbac:
  create: "true"
```

2. Run the commands:

```
helm3 repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm3 repo update
helm3 install ingress-controller ingress-nginx/ingress-nginx --namespace kube-system --values ingress-config.yaml
```

Kubernetes/ingress-nginx ingress controller `config.map` settings

To ensure that the IBM API Connect services have time to start, increase the *proxy-read-timeout* and *proxy-send-timeout* values, which are in seconds, in the `kubernetes/ingress-nginx` ingress controller *config.map* to at least the following:

- proxy-read-timeout: "240"
- proxy-send-timeout: "240"

Depending on your environment, you might need to increase these further if the IBM API Connect services do not start. If there is a load balancer in front of the worker nodes, then the load balancer configuration might also need to have extended timeouts.

System and Software Requirements

The system and software requirements are described in the Software Product Compatibility Reports. See Detailed system requirements for a specific product

Persistent Volume (PV) requirements

Review the persistent volumes (PVs) required by your API Connect installation.

The PV requirements for API Connect subsystems vary according to the number of subsystem nodes that are deployed, whether you are using two datacenter disaster recovery (2DCDR) Two data center warm-standby deployment on Kubernetes and OpenShift, and other subsystem specific options.

Management subsystem

Table 1. Management PV requirements

Deployment profile	Number of PVs required
Single node (n1)	4
Three node (n3)	10

Note: An additional 4 PVs are required for upgrade of the n3 deployment profile. Replaced PVs can be deleted after successful upgrade.

Gateway subsystem

Only required where `tokenManagementService` is enabled:

```
tokenManagementService:
  enabled: true
  storage:
    storageClassName: $STORAGE_CLASS
    volumeSize: 30Gi
```

Portal subsystem

Table 2. Portal PV requirements

Deployment profile	Number of PVs required
Single node (n1), one datacenter	6
Three nodes (n3), one datacenter	18
Single node (n1), two datacenter	9 per datacenter
Three nodes (n3), two datacenter	27 per datacenter
Single node (n1), two datacenter, where other datacenter is three node (n3)	13 in n1 datacenter, 23 in n3 datacenter
Three node (n3), two datacenter, where other datacenter is single node (n1)	30 in n3 datacenter, 15 in n1 datacenter

The amount of persistent storage space that is used by the developer portal depends on the following factors:

- The number of portal sites you create. Each portal site uses space in the `webVolumeClaimTemplate` and `databaseVolumeClaimTemplate` PVCs. 3 backups of each portal site are stored in the `backupVolumeClaimTemplate` PVC.
- The content that you add to your portal sites, such as custom images and documents that you attach to your sites.
- If debugging tools such as Drupal logs are enabled.

Note: In Openshift disk performance increases with disk size, for example, a 200 Gi PVC has better performance than a 30 Gi PVC. It is recommended that the `databaseVolumeClaimTemplate` is kept above 200 Gi to avoid performance issues.

Analytics subsystem

The number of PVs needed depends on if you disable local storage.

Table 3. Analytics PV requirements

Deployment profile	Local storage enabled	Local storage disabled (ingestion only)
Single node (n1)	2	1
Three node (n3)	6	3

To understand how much storage capacity your analytics subsystem needs, see Estimating storage requirements.

Load balancer configuration in a Kubernetes deployment

When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect Kubernetes deployment.

Before you begin

Note:

- This article does not apply to the IBM® DataPower® Gateway component that is a part of the IBM API Connect solution.
- This article applies to third-party software that IBM does not control. As such, the software may change and this information may become outdated.

About this task

API Connect can be deployed on a single node cluster. In this case the ingress endpoints are host names for which the DNS resolution points to the single IP address of the corresponding node hosting a particular subsystem, and no load balancer is required. For high availability, it is recommended to have at least a three node cluster. With three nodes, the ingress endpoints cannot resolve to a single IP address. A load balancer should be placed in front of an API Connect subsystem to route traffic.

Because it is difficult to add nodes once endpoints are configured, a good practice is to configure a load balancer even for single node deployments. With the load balancer in place, you can easily add nodes when needed. Add the node to the list of servers pointed to by the load balancer and the ingress endpoints defined during installation of API Connect can remain unchanged.

To support Mutual TLS communication between the API Connect subsystems, configure the load balancer with **SSL Passthrough** and **Layer 4** load balancing. In order for Mutual TLS to be performed directly by the API Connect subsystems, the load balancer should leave the packets unmodified, as is accomplished by Layer 4. Following is a description of the communication between the endpoints that are configured with Mutual TLS:

- API Manager (with the client certificate portal-client) communicates with the Portal Admin endpoint portal-admin (with the server certificate portal-admin-ingress)
- API Manager (with the client certificate analytics-ingestion-client) communicates with the Analytics Ingestion endpoint analytics-ingestion (with the server certificate analytics-ingestion-ingress)

Note: From v10.0.5.3, it is possible to disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see Enable JWT security instead of mTLS.

Set endpoints to resolve to the load balancer
: When configuring a load balancer in front of the API Connect subsystems, the ingress endpoints are set to host names that resolve to a load balancer, rather than to the host name of any specific node. For an overview of endpoints, see Deployment overview for endpoints and certificates.

Use these example topologies as a guideline to determine the best way to configure the load balancer for your deployment.

Procedure

- **Kubernetes cluster with wildcard DNS**
  In this example, API Connect is deployed to a Kubernetes cluster with three master and six worker nodes. The master nodes have the Kubernetes API server listening on port 6443. The Kubernetes API Server is used for communicating with the kubectl command line interface. Note that it is generally not necessary for the Kubernetes API server (on port 6443 in this example) to be exposed outside the cluster through the load balancer, this is just one possible setup shown here.

  The kubernetes/ingress-nginx ingress controller is deployed as a daemonset, so that every worker node in the cluster has an ingress controller pod listening on port 443. The API Connect subsystems (API Manager, Developer Portal, Analytics and Gateway) are all deployed on this same cluster.

  Note:
  - When you configure a load balancer in front of a Management subsystem, specify timeouts of at least 240 seconds. Note that large deployments might need larger values.

    The default timeout is typically 50 or 60 seconds, which is not long enough to avoid **`409 Conflict`** or **`504 Gateway Timeout`** errors. The **`409 Conflict`** error can occur when the time needed to complete an operation is sufficiently long that a second request gets issued.

    For example, to specify 240 seconds when using HAProxy as a load balancer, set **`timeout client`** and **`timeout server`** to **`240000`**.

    Best practice is to ensure that the same values are specified for the timeout settings for the load balancer and for the Kubernetes ingress controller. The ingress controller timeout settings are set in the ingress controller config.map. For more information, see the **`proxy-read-timeout`** and **`proxy-send-timeout`** settings in Kubernetes ingress controller prerequisites .
  - Ensure that the load balancer is configured to support the cipher **`TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`**.
    When the Developer Portal, as a client, calls back to platform-api-endpoint or consumer-api-endpoint of the management subsystem, it sends only a single cipher in the Client Hello. The cipher is **`TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)`**. The SSL Handshake will fail if this cipher is not configured (supported) on the client's load balancer.

  A host running HAProxy acts as the load balancer, with a configuration for proxies in the HAProxy configuration file such as:

```
defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  240000
        timeout server  240000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
```

```
frontend apiservers
        bind *:6443
        mode tcp
        option  tcplog
        option forwardfor
        default_backend k8s_apiservers

frontend ingress
        bind *:443
        mode tcp
        option  tcplog
        option forwardfor
        default_backend k8s_ingress

backend k8s_apiservers
        mode tcp
        option  tcplog
        option ssl-hello-chk
        option log-health-checks
        default-server inter 10s fall 2
        server cluster1-master-1 10.169.241.226:6443 check
        server cluster1-master-2 10.169.241.163:6443 check
        server cluster1-master-3 10.169.241.153:6443 check


backend k8s_ingress
        mode tcp
        option  tcplog
        option ssl-hello-chk
        option log-health-checks
        default-server inter 10s fall 2
        server cluster1-worker-1 10.169.241.142:443 check
        server cluster1-worker-2 10.169.241.150:443 check
        server cluster1-worker-3 10.169.241.188:443 check
        server cluster1-worker-4 10.169.241.196:443 check
        server cluster1-worker-5 10.169.241.168:443 check
        server cluster1-worker-6 10.169.241.156:443 check
```

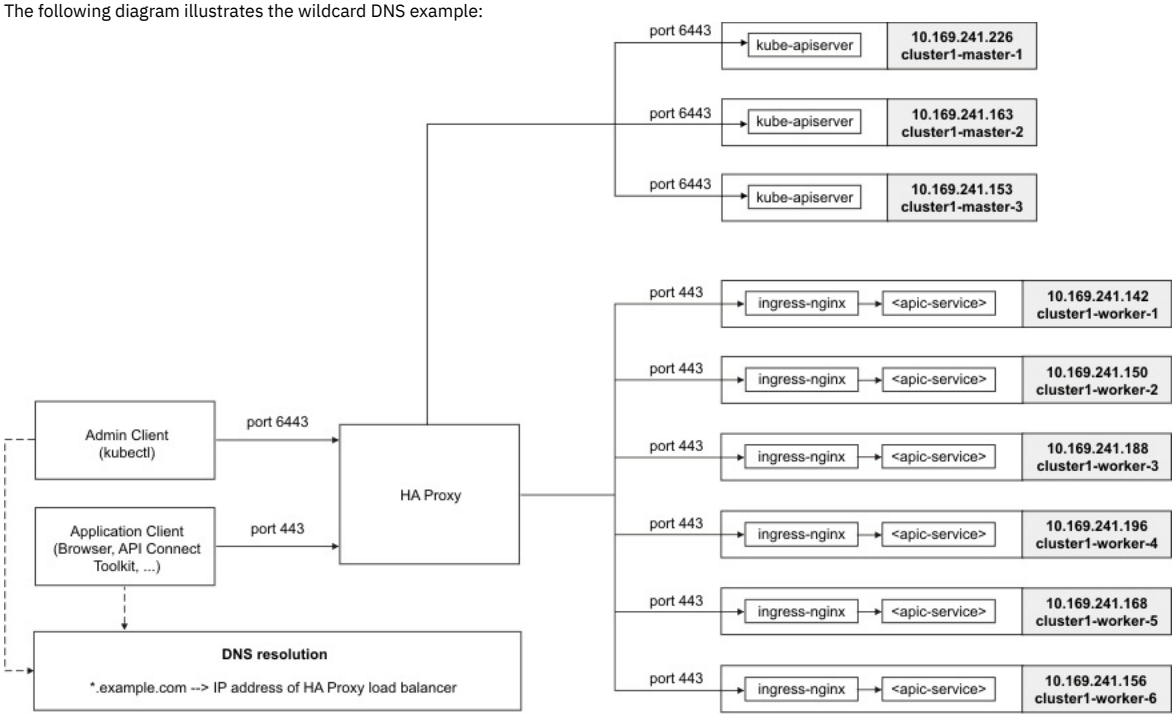
A wildcard DNS record allows the resolution of **`*.example.com`** to the IP address of the HAProxy host.

The ingress endpoints are defined in the CRs (custom resources) for each API Connect subsystem. During configuration of each subsystem, you define the endpoint by specifying a host name **`$STACK_HOST`** in the CR. For configuration instructions, see Installing the API Connect subsystems.

You can view the endpoint definitions in sample CRs:

- Installing the Management subsystem
- Installing the Developer Portal subsystem
- Installing the Analytics subsystem
- Installing the DataPower Gateway subsystem

The following diagram illustrates the wildcard DNS example:



- **Kubernetes cluster without wildcard DNS**
  This example uses the same topology as the previous one, except that there is no wildcard DNS resolution. Instead, individual DNS records are added, all pointing to the IP address of the load balancer. For example: **`portal.example.com`** and **`admin.portal.example.com`** resolve to the IP address of the load balancer.
- **Kubernetes cluster with `xip.io` or `nip.io`**
  This example uses the same topology as the first one, except that there is no specific DNS resolution configured and instead a service such as **`xip.io`** or **`nip.io`** provides wildcard DNS resolution. It is not recommended to use this approach for a production setup, as the ingress endpoints configured during installation of API

Connect would be tied to the IP address of the load balancer. This approach can be useful when configuring a test deployment in situations where access to DNS records is not practical or would introduce delay in the deployment.

Firewall requirements on Kubernetes

Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on Kubernetes.

Required Ports between zones

The following network diagram example helps to explain which ports must be configured in an API Connect network. Specific ports must be configured to enable the communication between the various zones, both public and private, in a network.

The ports specified in the diagram are default ports. Check your deployment to understand which communication, if any, is configured to use non-default ports.

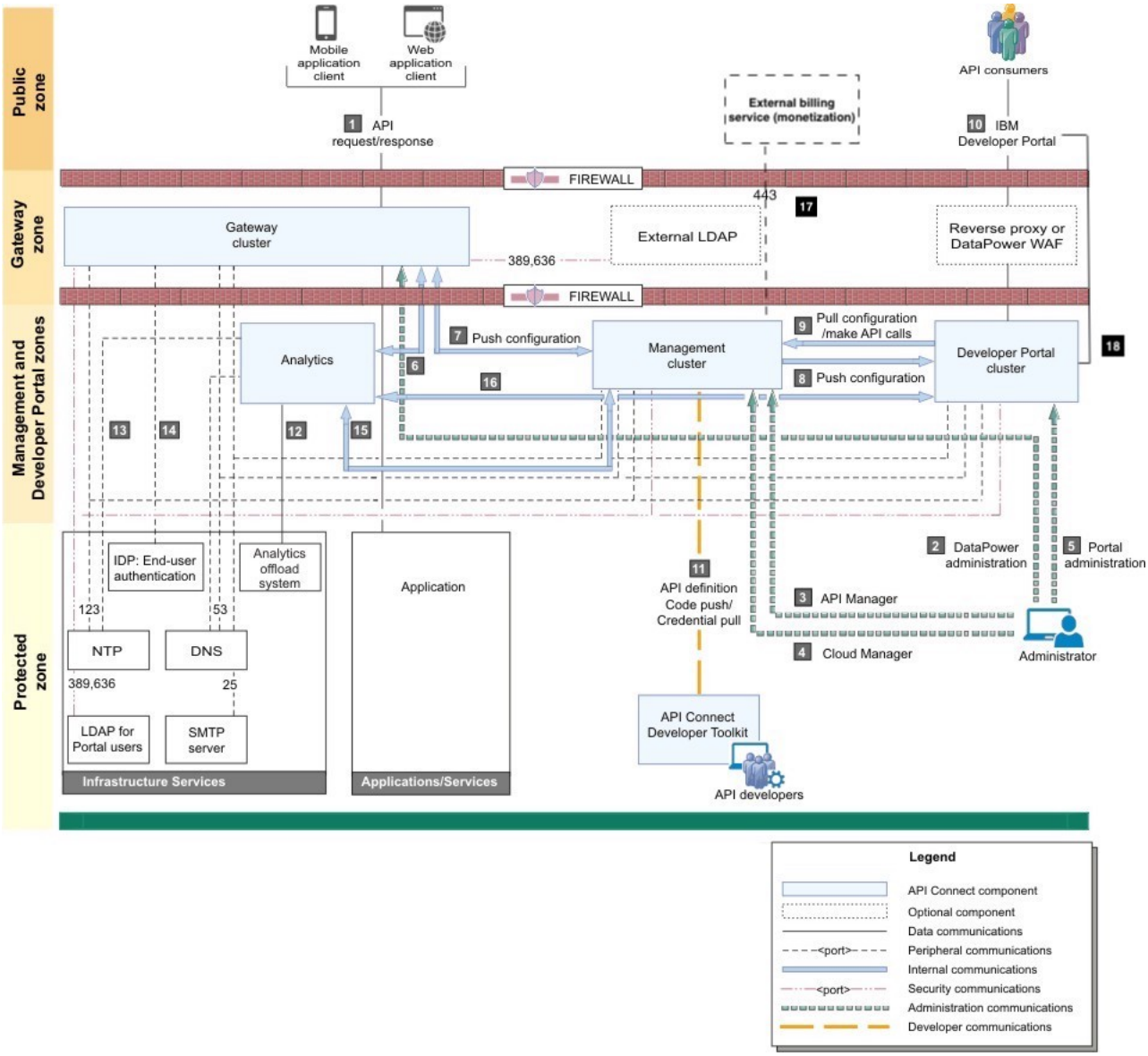


Table 1. Key for the network diagram example.
The following table lists the port numbers with a usage description.

	Usage description	Default port number
1	API request/response – Users invoking the provided APIs.	443 HTTPS from Public zone to Gateway zone
2	DataPower® administration – Internal operators who are managing the Gateway servers.	22 SSH, 9090 HTTPS from Protected zone to Gateway zone
3	API Manager – Internal business users who are defining and monitoring APIs.	443 HTTPS from Protected zone to Management zone
4	Cloud Manager – Internal operators who are administering the Cloud.	22 SSH, 443 HTTPS from Protected zone to Management zone
5	Developer Portal administration – Internal operators who are managing the Portal servers.	22 SSH, 443 HTTPS from Protected zone to Management zone

	Usage description	Default port number
6	Gateway servers post traffic to Analytics service.	443 HTTPS from Gateway servers to Analytics service
7	Push configuration – Management servers communicate bi-directionally with Gateway servers.	3000 and 443 HTTPS Management servers to and from Gateway servers for webhook delivery. Port 3000 is the default port for apic-gw-service, and this is the port Management uses to communicate with Gateway. Port 443 is the default port for the platform-api endpoint, which apic-gw-service uses to communicate with Management.
8	Push configuration/webhooks – Management servers push configuration and webhooks to the Developer Portal.	443 HTTPS Management servers to Developer Portal servers for webhook delivery
9	Pull configuration/make API calls – Developer Portal servers pull configuration and call REST APIs.	443 HTTPS from Developer Portal servers to Management servers within Management zone
10	Developer Portal – External developers who are accessing the Developer Portal.	443 HTTPS from Public zone to Developer Portal management zone. The reverse proxy/DataPower WAF for incoming web traffic to the Developer Portal cluster must be a transparent proxy - no modification of the portal URL, port, host name or path is allowed.
11	Push API definition to Management server. Pick up credential for microservice code push.	443 HTTPS from Protected zone to Management zone
12	Analytics offload	Port will depend on type of plugin and protocol used for the offload. Some possible protocols are: HTTP, HTTPS, TCP, UDP, KAFKA
13	Analytics accesses NTP	Standard NTP
14	Analytics access DNS	Standard DNS
15	Management service queries Analytics service	443 HTTPS within Management zone
16	The Portal service invokes an API (GET) on the Analytics service to retrieve data.	443 HTTPS within Management zone
17	External billing service – Management cluster connecting to external billing service (when configured for billing). If you are using Stripe as your external billing service, you must enable connections with the Stripe API at: https://stripe.com/files/ips/ips_api.json. You can also view the Stripe IP addresses at the following URL: https://stripe.com/docs/ips.	443 HTTPS from Management zone to Public zone
18	Developer Portal cluster must be able to access its own site endpoints.	443 HTTPS from Portal zone to Public zone

Firewall port requirements on Kubernetes

The following tables lists ports that must be open in both cluster and non-clustered deployments.

Table 2. Firewall port requirements common to all subsystems

Subsystem	Ports and description
Ports that must be open on all API Connect subsystems	The following ports must be open on the Management Server, Analytics, and Developer Portal subsystems, whether in a cluster or not. • 22 (outbound) Used for SFTP backups on Management and Portal subsystems • 53 (outbound) DNS • 123 (outbound) NTP • 443 (inbound and outbound) Used by all subsystems for communication with other subsystems • 44135 (inbound and outbound)

Each subsystem uses ports in addition to the ports in Table 2. See the following table.

Table 3. Additional firewall port requirements for each subsystem

Subsystem	Ports and description
Management Service	Management Service uses the ports in Table 2 plus: • 22 remote backup server port (configurable) If backups are configured to use another port, ensure that the port is open. • 161 (inbound) SNMP • LDAP server port (if using LDAP user registry), typically 389 (outbound) • 3007 (outbound) LDAP The Automated test behavior application must be able to invoke (outbound) any port that a test system might present an API on. It can be bound to a particular port (if it is the same in all environments) or to a range of ports (to cover all of the ports it might invoke).
Developer Portal	The Developer Portal uses the ports listed in Table 2, plus: • 22 - A remote backup server port (configurable) If backups are configured to use another port, ensure that the port is open. See Configuring portal database backups.
Analytics	The Analytics subsystem uses the ports listed in Table 2. Analytics port usage considerations: • 161 for SNMP is required for both non-clustered and clustered deployments • In a non-clustered deployment, no additional ports are required. • Analytics supports an optional configuration for offload of data. This configuration might require additional outbound ports to be open.

Subsystem	Ports and description
Gateway Server	The Gateway Server uses these ports in both non-clustered and clustered deployments: • 161 (inbound and outbound) SNMP • 162 (outbound) SNMP traps • 3000 (inbound) Gateway Service local port (configurable) • 5550 (inbound) XML management port (configurable) • 5554 (inbound) REST management port (if enabled; configurable) • 9022 (inbound) Gateway SSH (if enabled; configurable) • 9090 (inbound) Web GUI console (if enabled; configurable) • 9443 (inbound) Gateway local port (configurable)

Communications inside the Gateway cluster

There are a number of important points to note regarding the communications within the Gateway cluster.

- We advise that you use the same port for all Gateway servers within a cluster.
- Gateway servers communicate with each other to synchronize invocation counts.
- All Gateway servers in a Gateway cluster must be able to reach all of the other Gateway servers in the same Gateway cluster.
- Gateway servers in a Gateway cluster do not directly communicate with Gateway servers in a different Gateway cluster.
- All Gateway servers must be able to reach the management subsystem platform API endpoint, which was configured during the installation of your API Connect environment.

Ethernet interface usage

To separate network traffic, you can use two or more Ethernet interfaces on the DataPower appliance on which a Gateway server is installed. For example, you can use one interface for internal IBM API Connect communications, and another for processing incoming API calls.

Port requirement for Stripe billing service

All Management servers and Developer Portal servers must be able to communicate HTTPS content with the external billing service when billing is configured. If you are using Stripe as your external billing service, you must enable HTTPS communication on port 443 with the Stripe API: https://stripe.com/files/ips/ips_api.json.
Note: API Connect does not support using webhooks with Stripe. You can see a list of the Stripe IP addresses at: https://stripe.com/docs/ips.

Related concepts

- Installing and maintaining IBM API Connect

Related reference

- IBM API Connect Version 10 software product compatibility requirements

Planning your analytics deployment

Plan your API Connect analytics deployment by reviewing options for deployment profile and estimating required persistent storage space.

Deployment profile

Decide which deployment `profile` you want to use, the available profiles are:

- For Kubernetes and OpenShift (individual subsystem) installations: Analytics deployment profiles.
- For Cloud Pak for Integration and OpenShift top-level CR installations: API Connect deployment profiles.

For a scalable analytics deployment, choose a three replica profile. If you do not expect a high API transaction rate and plan to keep a limited amount of analytics data, then a one replica profile is sufficient.

Storage class

Before you install analytics, you must choose a storage class. **`Ceph`**, **`Block`**, and **`Local Volume`** are all supported; however, **`Local Volume`** is the most suitable for analytics. The analytics subsystem provides internal data replication and HA support, so the additional HA capabilities that **`Ceph`** or **`Block`** storage might give you are not needed. In addition, analytics requires a high throughput of disk I/O operations to successfully handle the analytics data load from the gateway. **`Local Volume`** is the best-performing storage class for disk I/O.

Note: GlusterFS and NFS storage are not supported for analytics. If you use either of these storage classes, you might encounter severe performance degradation, and possibly loss of data.

Storage type

Note: Storage type selection is available on three replica deployments only.
To make an informed choice on which storage type is best for you, first read and understand the OpenSearch concepts that are described in OpenSearch nodes, indices, shards, and replicas.
API Connect provides two OpenSearch storage types:

- **Shared storage**
  A single pod provides data storage and OpenSearch cluster management. Shared storage is the default option, and the only option on a one replica deployment.
- **Dedicated storage**
  OpenSearch data storage and cluster management functions are split, and run in separate pods on each worker node:
  - The `storage-os-master` pod runs an OpenSearch "manager eligible" node that manages the OpenSearch cluster, but does not store any API event data.
  - The `storage` pod runs an OpenSearch "data" node that stores API event data.

Dedicated storage facilitates horizontal scaling to support greater analytics data storage and throughput. Add worker nodes and increase the number of `storage` pod replicas to scale horizontally (you do not need to increase the number of `storage-os-master` replicas). Dedicated storage also provides greater stability and allows some OpenSearch configuration changes to be made without downtime.
You can change storage type after installation, see Dedicated storage and scaling up.

Disabling internal storage

The analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution if you are using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

When you disable internal storage for analytics data, the following microservices are disabled:

- `osinit`
- `storage`

Important:
If you disable local storage, the analytics views in your API Connect UIs remain accessible, but are empty.

Configure local storage disablement before installation. It is not possible to disable or re-enable internal storage after installation.

You can enable offload of analytics data to third-party systems after installation.

For the steps to disable internal storage, see Disable local storage.

Persistent storage space

Based on your expected API transaction rate, you can estimate how much storage space your analytics subsystem requires. See Estimating storage requirements.

Inter-subsystem communication security

By default the network communication from the management and gateway subsystems, to the analytics subsystem uses Kubernetes ingresses or OpenShift routes, and is secured with mTLS. You can configure alternative network communication options if your environment requires. For more information, see Network requirements for inter-subsystem communication.

Kubernetes operating map counts

When the analytics service is configured to store data, it uses OpenSearch, which requires map counts higher than the operating system defaults. The minimum recommended value is `262144`. Unless you plan to disable local storage, increase the default map count on every Kubernetes worker node:

1. To change the map counts on the live system, run the following command on every Kubernetes node:

   ```
   sudo sysctl -w vm.max_map_count=262144
   ```

2. To persist this change when node restarts occur, add the following setting to the /etc/sysctl.conf file:

   ```
   vm.max_map_count = 262144
   ```

For more information, see Important settings in the OpenSearch documentation.

PV requirements

The number of PVs needed depends on if you disable local storage.

Table 1. Analytics PV requirements

Deployment profile	Local storage enabled	Local storage disabled (ingestion only)
Single node (n1)	2	1
Three node (n3)	6	3

To understand how much storage capacity your analytics subsystem needs, see Estimating storage requirements.

Firewall requirements

The analytics firewall requirements are documented here: Firewall requirements

Key Concepts: Cert-manager, Issuers, and Secrets

By default API Connect uses an open source product that is called cert-manager to handle the issuing and renewal of the certificates that are used by API Connect. The cert-manager has its own Kubernetes pods and runs in its own namespace. The cert-manager adds some additional resources to the Kubernetes environment. The API Connect administrator needs to be familiar with these additional resources:

Issuers

Issuers play the part of certificate authorities (CAs) in the local environment and issue the certificates. Issuers reside in the API Connect namespace. There are two types of Issuers:

- Self-Signed - This issuer provides the self-signed certificates, for example a root CA.
- CA - This issuer can be either the root certificate authority or an intermediate certificate authority.

Here is an example of the issuers that can be found on the management subsystem:

```
kubectl get issuers -n <management namespace>
NAME                 READY   AGE
abc-management-ca    True    96d
selfsigning-issuer   True    96d
```

- `selfsigning-issuer` This is a SelfSigned Issuer that provides the root certificate for all default certificates that are used in API Connect.
- `abc-management-ca` This is a CA Issuer that signs all the end-entity certificates that are used for communication between the management subsystem pods. It is in effect an intermediate CA between the root (`selfsigning-issuer`) and the end-entity certificates.

Certificates

A certificate is a Kubernetes resource that contains a pointer to a Kubernetes secret. The Kubernetes secret contains the actual TLS certificate. Certificate objects contain other information such as the issuer, expiry date, and status. Run `kubectl get certificate` to see the certificates on an API Connect subsystem:

```
kubectl get certificate -n <management namespace>
NAME                                    READY   SECRET                                  AGE
abc-management-ca                       True    abc-management-ca                       96d
abc-management-client                   True    abc-management-client                   96d
abc-management-natscluster-mgmt         True    abc-management-natscluster-mgmt         96d
abc-management-server                   True    abc-management-server                   96d
...
```

The first certificate in the list `abc-management-ca` is the intermediate CA certificate that is used for signing the other certificates that are shown in the output. If you describe this certificate you can see that the `selfsigning-issuer` is its issuer:

```
kubectl describe certificate abc-management-ca -n <management namespace>
...
Spec:
  ...
  Issuer Ref:
    Kind:  Issuer
    Name:  selfsigning-issuer
```

Describing the other management subsystem certificates, observe that the `abc-management-ca` certificate is the issuer:

```
kubectl describe certificates abc-management-client -n <management namespace>
...
Spec:
  ...
  Issuer Ref:
    Kind:  Issuer
    Name:  abc-management-ca
```

Key points:

- The certificates with '-ca' on the end are all CA Certificates, and they sign all the other subsystem certificates whose names do not end with '-ca'.
- The CA certificates are also known as Issuer certificates.
- Certificates that CA certificates sign are known as end-entity certificates.
- When a CA certificate is updated, all end-entity certificates that it signs must also be updated.

For a complete list of the certificates in an API Connect installation, see API Connect certificates reference.

Secrets

Secrets are the Kubernetes resource for storing credentials. In API Connect, secrets are where the TLS certificates are stored. Every Kubernetes certificate object has a corresponding secret of the same name. The Kubernetes certificate contains a property that refers to its corresponding secret:

```
kubectl describe certificates abc-management-client -n <management namespace>
...
Spec:
  ...
  Secret Name:        abc-management-client
```

The default output of `kubectl get certificate` includes a column that is labeled `SECRET`, which shows the secret name.

```
kubectl get certificate -n <management namespace>
NAME                                    READY   SECRET                                  AGE
abc-management-ca                       True    abc-management-ca                       96d
abc-management-client                   True    abc-management-client                   96d
abc-management-natscluster-mgmt         True    abc-management-natscluster-mgmt         96d
abc-management-server                   True    abc-management-server                   96d
...
```

The `kubectl describe` output of a secret shows that it contains the TLS certificates:

```
kubectl describe secret abc-management-client -n <management namespace>
...
Name:         abc-management-client
Namespace:    default
```

```
Labels:       app.kubernetes.io/instance=abc-management
              app.kubernetes.io/managed-by=ibm-apiconnect
              app.kubernetes.io/name=abc-management-client
Annotations:  cert-manager.io/alt-names:
              cert-manager.io/certificate-name: abc-management-client
              cert-manager.io/common-name: abc-management-client
              cert-manager.io/ip-sans:
              cert-manager.io/issuer-group:
              cert-manager.io/issuer-kind: Issuer
              cert-manager.io/issuer-name: abc-management-ca
              cert-manager.io/uri-sans:

Type:  kubernetes.io/tls

Data
====
ca.crt:   1107 bytes
tls.crt:  1139 bytes
tls.key:  1675 bytes
```

The full TLS certificate strings can be seen with:

```
kubectl get -o yaml secret <secret name> -n <namespace>
```

For more details about cert-manager, and a full list of all the certificates that are created and used in an API Connect deployment, see API Connect certificate reference.

Certificates in a Kubernetes environment

Use of cert-manager is recommended for managing certificates in a Kubernetes environment.

When deploying on Kubernetes, choose one of the following methods for creating internal certificates:

- Do not specify any internal certificates. In this case, the operator generates all of the internal certificates without using a certificate manager. The CR definition to enable this option is:

  ```
  microServiceSecurity: custom
  ```

- Specify some, or all, of the internal certificates yourself without using a certificate manager. In this case, any certificates that you do not create are created by the operator. You retain control over the certificates that you create. The CR definition to enable this option is:

  ```
  microServiceSecurity: custom
  ```

- Configure cert-manager to generate the internal certificates. In this case, cert-manager creates and manages the certificates. The CR definition to enable this option is:

  ```
  microServiceSecurity: certManager
  ```

  To use cert-manager for generating the certificates, download and install cert-manager v1.11.5 from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5 and create an Issuer CR in the same namespace where you will install API Connect; for example:

  ```
  apiVersion: cert-manager.io/v1
  kind: Issuer
  metadata:
    name: selfsigning-issuer
  spec:
    selfSigned: {}
  ```

  In the CR for the API Connect Management subsystem, specify a **certManagerIssuer**. The **certManagerIssuer** can be configured to be a self-signing issuer, as in the following example:

  ```
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  ```

  API Connect supports any other kind of Issuer supported by cert-manager, in addition to the self signed type.

In all the CRs, certs are expected at the level of any endpoint. For example, for the cloud admin endpoint of the Management subsystem you could have:

```
cloudManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: admin.example.com
      secretName: cm-endpoint
```

The previous entry lets cert-manager automatically generate a **cm-endpoint** TLS secret using the **ingress-issuer**. One way to define the **ingress-issuer** is as follows:

```
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
```

```
kind: Certificate
metadata:
  name: ingress-ca
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
spec:
  ca:
    secretName: ingress-ca
```

Another way to configure the cloud admin endpoint would be with:

```
 cloudManagerEndpoint:
    hosts:
    - name: admin.example.com
      secretName: my-cm-endpoint
```

Where **my-cm-endpoint** is the name of a secret that you previously created, and is present in the same namespace as the subsystem, and contains TLS certificate files for termination of the endpoint.

Creating secrets and generating certificates

There is a number of "secrets" for which you can provide a name in the CRs:

- Portal admin secret.

  ```
  portal:
    admin:
      secretName:
  ```

- Analytics ingestion secret.

  ```
  analytics:
    ingestion:
      secretName: analytics-ingestion-client
  ```

- APIC Gateway Service TLS secret. Template entry from gateway_cr_v6.yaml (in the file name, "v6" refers to the API Gateway service):

  ```
    apicGatewayServiceTLS:
      secretName:
  ```

- APIC Gateway Peering TLS secret. Template entry from gateway_cr_v6.yaml (in the file name, "v6" refers to the API Gateway service):

  ```
    apicGatewayPeeringTLS:
      secretName:
  ```

You can create manually create secrets that contain the tls.crt, tls.key, and tls.ca files for a certificate. You can then refer to them by name in their subsystem CR. However, a simpler way to have those secrets created is by using cert-manager. The following example shows how to generate those certificates with cert-manager:

```
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
```

```
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
spec:
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
spec:
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
spec:
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
spec:
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

So with the previous example, in the subsystem CRs (for Management and Gateway) the following values could be set:

-
```
portal:
  admin:
    secretName: portal-admin-client
```

-
```
analytics:
  ingestion:
    secretName: analytics-ingestion-client
```

-
```
apicGatewayServiceTLS:
  secretName: gateway-service
```

- `apicGatewayPeeringTLS:`
  `secretName: gateway-peering`

Also, the Client certificate should be signed by same CA as the one for the endpoint certificates, and the server side should be configured with the expected Subject DN for the client. For example:

- The `portalAdminSecret` certificate and the certificate used to terminate `portalAdminEndpoint` must have the same CA (in the example the CA is `ingress-ca` for both). Also, the `adminClientSubjectDN` in the Portal subsystem CR should be set to the Subject DN of `portal.admin.secretName`. For example:

  ```
  adminClientSubjectDN: CN=portal-admin-client,O=cert-manager
  ```

- Similarly, the `analyticsIngestionSecret` and the Analytics ingestion endpoint certificate should share the same CA. The `ingestion.clientSubjectDN` should be set to the Subject DN of the `analytics.ingestion.secretName` certificate.

  ```
  ingestion:
    endpoint:
      annotations:
        cert-manager.io/issuer: ingress-issuer
      hosts:
      - name: ai.example.com
        secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
  ```

Important: When installing more than one Analytics subsystem in the same Kubernetes namespace, all secret names must be unique.

Creating TLS secret using openssl

Use of the cert-manager is recommended for managing certificates. However, you can choose not to use cert-manager, and instead create certificates manually. Requirements for manually created certificates:

- Extended Key Usage (EKU), either serverAuth or clientAuth depending upon the type of certificate. Certificates of type *Server* must have an Extended Key Usage with serverAuth purpose. Certificates of type *Client* must have an Extended Key Usage with clientAuth purpose.
- Subject Alternative Name (SAN) for the required hosts
- If a certificate is signed by an internal or custom CA, include the full chain in the end certificate. If you omit the full chain, then a user who uses openSSL to access the endpoint will see the following error: `error:num=20:unable to get local issuer certificate`
  The following example shows the full chain for an end certificate (signed by a custom CA):

  ```
  -------BEGIN CERTIFICATE -----
  Cert contents for end-cert
  --------END CERTIFICATE ----------
  -------BEGIN CERTIFICATE -----
  Cert contents for Intermediate-CA
  --------END CERTIFICATE ----------
  -------BEGIN CERTIFICATE -----
  Cert contents for Root CA
  --------END CERTIFICATE ----------
  ```

The following steps can be used as an alternative to using cert-manager. If you choose to create your certificates this way, you lose some of the management features of cert-manager. For example, you must manually update your certificates when their expiration date gets near.

1. Generate the custom certificate with the appropriate EKU and SAN. You will need to obtain the private key, public certificate, and CA certificates in non-password-protected PEM format for the custom certificate. Following is an example for how to generate a certificate (platform-api-example) with an EKU serverAuth and SAN using openssl:

   ```
   openssl x509 -req -days 360 -in platform-api-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out
   platform-api-cert -sha256
   -extfile <(cat /etc/ssl/openssl.cnf <(printf
   "\n[SAN]\nsubjectAltName=DNS:fqdn.myserver.com\nextendedKeyUsage=serverAuth"))
   -extensions SAN
   ```

   where
   - `DNS:fqdn.myserver.com` is the fully qualified domain name of the endpoint the certificate applies to.
   - `platform-api-example.csr` is the file name for the certificate signing request

   Following is an example for how to generate a certificate (portal-client) with an EKU clientAuth and SAN using openssl:

   ```
   openssl x509 -req -days 360 -in portal-client-example.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -out
   portal-client-cert
   -sha256 -extfile <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nkeyUsage=critical,
   digitalSignature,keyEncipherment\nextendedKeyUsage = clientAuth\nbasicConstraints=critical,
   CA:FALSE\nsubjectKeyIdentifier=hash\n")) -extensions SAN
   ```

2. Create a Kubernetes secret containing tls.cert, tls.key, and ca.cert.

Setting an encryption-secret for the management database using openssl

You can optionally set the encryption-secret for the management database. If you do not set one, one is automatically generated for you.

1. The encryption-secret is a secure random bytes password used for field level encryption in the management database. You can generate 128 random bytes using the following command in openssl:

   ```
   openssl rand -out /path/to/secret/encryption-secret.bin 128
   ```

2. Use kubectl to put the generated secret into `management-encryption-key`:

   ```
   kubectl create secret generic management-encryption-key -n <namespace>
         --from-file=/path/to/secret/encryption_secret.bin
   ```

   where `<namespace>` is the namespace where the Management CR is going to be created)

3. Make sure that in management_cr.yaml, encryptionSecret.secretName points to the name of the secret created in the previous step.

Important: Back up the encryption-secret.bin file to a safe location. If this file lost it will not be possible to access the management database or its backups.

- **Custom certificates on Kubernetes**
  You can use custom certificates in your Kubernetes deployment. You can optionally generate your own custom certificates.

Custom certificates on Kubernetes

You can use custom certificates in your Kubernetes deployment. You can optionally generate your own custom certificates.

To deploy API Connect with Custom Certificates, some additional preparation and steps are required before any subsystem CRs are applied into the Cluster.

You can provide existing custom certificates that you already own, for example DigiCert certificates, or generate new custom certificates using a package such as Cert-Manager.

- If you have your own custom certificates, see Configuring custom certificates before installation.
- If you do not already have your own custom certificates, and would like to use Cert-Manager to generate custom certificates, see Generating custom certificates
- To view a list of certificates used by API Connect, see Custom certificates reference

- **Custom certificates reference**
  API Connect supports custom certificates for each subsystem, and for internal communications.
- **Generating custom certificates**
  You can generate custom certificates.
- **Generating custom certificates using cert-manager in a two data center deployment**
  You can generate custom certificates in a two data center disaster recovery deployment on Kubernetes.
- **Configuring custom certificates before installation**
  Configure custom certificates for your API Connect deployment before you begin the installation procedures.
- **Converting to custom front-end/ingress certificates after deployment**
  Convert a front-end certificate generated by cert-manager to a custom certificate for an existing IBM® API Connect deployment.

Custom certificates reference

API Connect supports custom certificates for each subsystem, and for internal communications.

Table 1. Certificates for Management Subsystem:

Certificate	Type	Notes
`analytics-ingestion-client`	Common Subsystem Communication	Must be signed by the same CA as `analytics-ai-endpoint` Certificate of Analytics Subsystem. For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=a7s-ingestion-client`, or they could both be `CN=a7s-ingestion-client, O=cert-manager`, but they must be identical.
`portal-admin-client`	Common Subsystem Communication	Must be signed by the same CA as `portal-admin` Certificate of Portal Subsystem. For a two data center deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=portal-admin-client`, or they could both be `CN=ptl-adm-client, O=cert-manager`, but they must be identical.
`gateway-client-client`	Common Subsystem Communication	Must be signed by the same CA as `gwv5-management-endpoint` and/or `gwv6-management-endpoint` of Gateway Subsystem
`cm-endpoint`	External Frontend/Ingress	
`apim-endpoint`	External Frontend/Ingress	
`api-endpoint`	External Frontend/Ingress	
`consumer-endpoint`	External Frontend/Ingress	
`hub-endpoint`	External Frontend/Ingress	
`turnstile-endpoint`	External Frontend/Ingress	

Note:
To generate the certificates for the endpoints used by the Automated testing behavior endpoints (`hub-endpoint` and `turnstile-endpoint`), add the following statements to the `custom-certs-external.yaml`:

```
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: hub-endpoint
  labels: {
    app.kubernetes.io/instance: "management",
```

```
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "hub-endpoint"
  }
spec:
  commonName: hub-endpoint
  secretName: hub-endpoint
  dnsNames:
  - hub.example.com
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: turnstile-endpoint
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "turnstile-endpoint"
  }
spec:
  commonName: turnstile-endpoint
  secretName: turnstile-endpoint
  dnsNames:
  - turnstile.example.com
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

Table 2. Certificates for Analytics Subsystem

Certificate	Type	Notes
`analytics-ai-endpoint`	External Frontend/Ingress	Must be signed by the same CA as `analytics-ingestion-client` Certificate of Management Subsystem

Table 3. Certificates for Gateway Subsystem

Certificate	Type	Notes
`gwv5-endpoint`	External Frontend/Ingress	
`gwv5-management-endpoint`	External Frontend/Ingress	Must be signed by the same CA as `gateway-client-client` Certificate of Management Subsystem
`gwv6-endpoint`	External Frontend/Ingress	
`gwv6-management-endpoint`	External Frontend/Ingress	Must be signed by the same CA as `gateway-client-client` Certificate of Management Subsystem

Table 4. Certificates for Portal Subsystem

Certificate	Type	Notes
`portal-admin`	External Frontend/Ingress	Must be signed by the same CA as `portal-admin-client` Certificate of Management Subsystem
`portal-web`	External Frontend/Ingress	

Table 5. Internal Certificates

Certificate	Type (CA/Server/Client)	Subsystem	Notes
`caCertificate`	CA	Management, Analytics, Portal	
`clientCertificate`	Client	Management, Analytics, Portal	

`serverCertificate`	Server	Management, Analytics, Portal	Portal DNS names required
			`*.<namespace>`
			`*.<namespace>.svc`
			`*.<instance name>-server.<namespace>.svc`
			`<instance name>-server`
			`*.<instance name>-<site name>-db-all.<namespace>.svc`
			`*.<instance name>-<site name>-www-all.<namespace>.svc`
			`*.<instance name>-<site name>-db-all.<namespace>.svc.cluster.local`
			`*.<instance name>-<site name>-www-all.<namespace>.svc.cluster.local`
			`*.<namespace>.svc.cluster.local`
			`<instance name>-db`

`dbServerCertificate`	Server	Management	
v10.0.6 only: `pgBouncerServerCertificate`	Server	Management	
v10.0.6 only: `PGOTLSCertificate`	Server	Management	
`NATSTLSCertificate`	Server	Management	
`dbClientPostgres`	Client	Management	

Certificate	Type (CA/Server/Client)	Subsystem	Notes
v10.0.6 only: `dbClientReplicator`	Client	Management	
v10.0.6 only: `dbClientPgbouncer`	Client	Management	
`dbClientApicuser`	Client	Management	
v10.0.6 only: `dbClientPrimaryuser`	Client	Management	

Several certificates as noted above are required to be signed by the same CA as another certificate. For example, `portal-admin-client`, and `portal-admin`. This means that if the `portal-admin-client` certificate were to be customized, then the `portal-admin` certificate must also be customized, and signed by the same CA as `portal-admin-client`. To ensure that pairs of certificates like these are signed by the same CA, the Issuer for each certificate must be the same.

Generating custom certificates

You can generate custom certificates.

About this task

If you do not already have your own custom certificates that you can provide for the installation, you can manually prepare Certificate definitions for Cert-Manager to generate the required Certificates for you.

Note: If a certificate is signed by an internal or custom CA, include the full chain in the end certificate. If you omit the full chain, then a user who uses openSSL to access the endpoint will see the following error: `error:num=20:unable to get local issuer certificate`
The following example shows the full chain for an end certificate (signed by a custom CA):

```
-------BEGIN CERTIFICATE -----
Cert contents for end-cert
--------END CERTIFICATE ----------
-------BEGIN CERTIFICATE -----
Cert contents for Intermediate-CA
--------END CERTIFICATE ----------
-------BEGIN CERTIFICATE -----
Cert contents for Root CA
--------END CERTIFICATE ----------
```

Procedure

1. Create a Certificate Authority (CA) and an Issuer for Signing
   a. Use the following YAML definition to generate a CA named `ingress-ca`, and an Issuer named `ingress-issuer` using that CA. Place the following definition into a file such as `ingress-ca.yaml`:

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: ingress-ca
spec:
    secretName: ingress-ca
    commonName: "ingress-ca"
    usages:
    - digital signature
    - key encipherment
    - cert sign
    isCA: true
    issuerRef:
        name: selfsigning-issuer
        kind: Issuer
    duration: 87600h # 10 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
name: ingress-issuer
spec:
    ca:
        secretName: ingress-ca
```

   b. Apply this file into the cluster:

```
kubectl apply -f ingress-ca.yaml -n <namespace>
```

   This will create a CA Certificate named `ingress-ca`, and also a Secret named `ingress-ca`. This CA will be used to sign the rest of the custom certificates.

   c. Verify the presence of both:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

2. Generating common subsystem communication certificates with Cert-Manager for management CR.
   a. Use the following YAML definition to generate the following Common Certificates and Secrets: `analytics-ingestion-client`, and `portal-admin-client`. Place it into a file, for example, `common-certs.yaml`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: analytics-ingestion-client
spec:
    commonName: analytics-ingestion-client
    secretName: analytics-ingestion-client
    issuerRef:
        name: ingress-issuer
    usages:
    - "client auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: portal-admin-client
spec:
    commonName: portal-admin-client
    secretName: portal-admin-client
    issuerRef:
        name: ingress-issuer
    usages:
    - "client auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
---
```

Note that the **issuerRef** uses the Issuer created in Step 1. These certificates must be signed by the CA associated with that Issuer, **ingress-ca**.

b. Apply **common-certs.yaml** into the cluster:

```
kubectl apply -f common-certs.yaml -n <namespace>
```

This will create Client Certificates and Secrets named **analytics-ingestion-client**, and **portal-admin-client**.

c. Verify the presence of all of these certificates and secrets:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

3. Generating Front-end/Endpoint Certificates with Cert-Manager
   a. Generate server certificates.
   For each Endpoint in a subsystem CR, you can use the following YAML definition to generate a Server Certificate:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: <endpoint-host.secretName>
spec:
    commonName: <endpoint.host.secretName>
    secretName: <endpoint.host.secretName>
    dnsNames:
    - <endpoint.host.Name>
    issuerRef:
        name: ingress-issuer
    usages:
    - "server auth"
    - "signing"
    - "key encipherment"
    duration: 17520h # 2 years
    renewBefore: 720h # 30 days
    privateKey:
      rotationPolicy: Always
```

Replace **<endpoint.host.secretName>** with the **secretName** in the **host** block of the **Endpoint** in the CR. For example, if the **Endpoint** in the CR is:

```
cloudManagerEndpoint:
    hosts:
    - name: admin.example.com
      secretName: cm-endpoint
```

The Certificate YAML should be updated as follows:

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
name: cm-endpoint
spec:
    commonName: cm-endpoint
    secretName: cm-endpoint
    dnsNames:
    - admin.example.com
    issuerRef:
```

```
      name: ingress-issuer
  usages:
  - "server auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

Note that the **issuerRef** uses the Issuer created in Step 1. This certificate will be signed by the CA associated with that Issuer, **ingress-ca**.

Place the edited yaml into a file, e.g. **<*subsystem*>-server.yaml**.

b. Apply this file into the cluster:

```
kubectl apply -f <subsystem>-server.yaml.
```

This will create a Server Certificate named **<*subsystem*>-server**, and also a Secret named **<*subsystem*>-server**.

c. Check for the presence of both with:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

d. Repeat Step 3.a, step 3.b and step 3.c for each subsystem.
In each case, the Certificates must use the same Issuer created in Step 1. This ensures that all custom certificates are signed by the same CA Certificate.

Note that the **issuerRef** uses the Issuer created in Step 1. These certificates must be signed by the CA associated with that Issuer, **ingress-ca**.

e. Place the YAML definitions for each Certificate into a YAML file, for example, **endpoint-certs.yaml**. Apply this file into the cluster:

```
kubectl apply -f endpoint-certs.yaml -n <namespace>
```

This will create Client Certificates and Secrets for each Endpoint named **<endpoint.host.secretName>**.

f. Check for the presence of all of these with:

```
kubectl get certificate -n <namespace>
kubectl get secret -n <namespace>
```

4. Choose one of the following actions, based on whether or not you are customizing internal API certificates:
   - If you do not plan to customize internal certificates, continue with Configuring custom certificates before installation.
   - If you want to generate some, or all, of the internal certificates, continue with the next step.

   Note: It is not recommended to customize Internal certificates unless you have a specific requirement to do so.
5. Generate custom internal certificates using custom-certs-internal.yaml.
   Each of the Internal certificates listed in Custom certificates reference have differing strict requirements for Common Names and DNS Names (i.e. Subject Alternative Names).

   To generate custom certificates that meet these requirements, use the provided custom-certs-internal.yaml file, which can be found in the **helper_files** installation archive. For more information, see Obtaining product files.

   This file contains Certificate YAML definitions for all Internal certificates, CA, and Issuer that can be customized for each subsystem of IBM API Connect. The Certificate definitions file is already populated with the recommended default values.

   a. When using custom internal certificates, you are required to specify a site name. The site name is used as the identifier for the PostgreSQL database Cluster used by API Connect. The site name chosen is added to the Management Subsystem CR YAML file before installing the Management Subsystem.
   In addition, this name is also required for use with the custom internal certificates, so a site name should be chosen now before proceeding.

   b. **v10.0.6 only:**Open custom-certs-internal.yaml in a text-editor, and make the following changes:
      i. Search for the **pg-bouncer-server-certificate** certificate definition. Replace **commonName** and **dns-names** with the following values:

      - ```
        commonName: management-pgo_cluster_id-postgres-pgbouncer
        ```
      - ```
        dnsNames:
          - "*.<namespace>"
          - "*.<namespace>.svc"
          - "management-pgo_cluster_id-postgres-pgbouncer.<namespace>.svc"
          - "management-pgo_cluster_id-postgres-pgbouncer"
        ```

      ii. Find and replace each occurrence of **<*namespace*>** with the desired namespace for the deployment. This must be done even if the **default** namespace is to be used.
      iii. Also in custom-certs-internal.yaml, replace each occurrence of **pgo_cluster_id** with the chosen site name described in Step 5.a.

   c. Apply the file to the Cluster:

   ```
   kubectl apply -f custom-certs-internal.yaml -n <namespace>
   ```

6. When all desired custom certificates are created and applied into the Kubernetes Cluster as **Certificates** and **Secrets**, update the Subsystem CR YAML files to use the custom certificates. See Configuring custom certificates before installation.

Generating custom certificates using cert-manager in a two data center deployment

You can generate custom certificates in a two data center disaster recovery deployment on Kubernetes.

About this task

You can provide your own custom certificates when you deploy IBM API Connect, if required, instead of having the application generate and manage them for you. In most deployment cases to use the built in default Cert-Manager integration is the preferred method.

When you use the custom option, the use of Cert-Manager is the recommended method for generating custom certificates if that fits your business needs. However, use of Cert-Manager is not required, and custom certificates can be generated in any way that suit your needs.

The following example of providing custom certificates uses Cert-Manager as a model for generating them. For generating custom certificates, the minimum supported version of cert-manager is v1.5.1.

Procedure

1. Set **KUBECONFIG** for the target cluster:

   ```
   export KUBECONFIG=<path_to_cluster_config_YAML_file>
   ```

   Example path:

   ```
   /Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml
   ```

2. Install cert-manager version 1.5.1 or later :
   a. Download v1.11.5 from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5.
   b. Install cert-manager. Do not specify a namespace:

   ```
   kubectl apply -f cert-manager.yaml
   ```

   c. Check the status of the **cert-manager** pods:

   ```
   kubectl -n cert-manager get po
   ```

   Wait for **cert-manager** pods to enter **Running 1/1** status before proceeding. There are 3 **cert-manager** pods in total.

3. Set up the customized external certificates:
   For quick setup, you can use the provided yaml files that are included in the **helper_files** that you expanded in Deploying operators and cert-manager. The yaml for the active data center (DC1) is called **custom-certs-external-dc1.yaml** and the yaml for the warm-standby data center (DC2) is called **custom-certs-external-dc2.yaml**.

   a. Open **custom-certs-external-dc1.yaml** and **custom-certs-external-dc2.yaml** in a text-editor. Find and replace each occurrence of **example.com** with the desired ingress subdomain for the API Connect stack.
   b. Ensure that the management CR for DC1 is ready for applying but not yet applied, make a note of the **hostname** of the site.
   c. In **custom-certs-external-dc1.yaml** find **spec.dnsNames**
   For example, in DC1:

   ```
   apiVersion: cert-manager.io/v1
   kind: Certificate
   metadata:
     name: dc1-mgmt-replication
   spec:
     commonName: dc1-mgmt-replication
     secretName: dc1-mgmt-replication
     dnsNames:
     - <INSERT SITE HOSTNAME HERE. THIS WILL BE THE SAME VALUE AS IN THE MANAGEMENT CR spec.multiSiteHA.hosts.name for
   site 1>
     issuerRef:
       name: ingress-issuer
     usages:
     - "signing"
     - "key encipherment"
     duration: 17520h # 2 years
     renewBefore: 720h # 30 days
     privateKey:
       rotationPolicy: Always
   ```

   d. Replace the line that begins **- <INSERT SITE HOSTNAME HERE** with *- hostname of the site*.
   e. Ensure that the management CR for DC2 is ready for applying but not yet applied, make a note of the **hostname** of the site.
   f. In **custom-certs-external-dc2.yaml** find **spec.dnsNames** and replace the line that begins **- <INSERT SITE HOSTNAME HERE** with *- hostname of the site*.
   g. Ensure that the portal CR for DC1 is ready for applying but not yet applied, make a note of the **hostname** of the site.
   h. In **custom-certs-external-dc1.yaml** find **spec.dnsNames**
   For example, in DC1:

   ```
   apiVersion: cert-manager.io/v1
   kind: Certificate
   metadata:
     name: dc1-ptl-replication
   spec:
     commonName: dc1-ptl-replication
     secretName: dc1-ptl-replication
     dnsNames:
     - <INSERT SITE HOSTNAME HERE. THIS WILL BE THE SAME VALUE AS IN THE PORTAL CR spec.multiSiteHA.hosts.name for site
   1>
       issuerRef:
       name: ingress-issuer
     usages:
     - "signing"
     - "key encipherment"
     duration: 17520h # 2 years
     renewBefore: 720h # 30 days
     privateKey:
       rotationPolicy: Always
   ```

i. Replace the line that begins `- <INSERT SITE HOSTNAME HERE>` with `- hostname of the site`.
j. Ensure that the portal CR for DC2 is ready for applying but not yet applied, make a note of the `hostname` of the site.
k. In `custom-certs-external-dc2.yaml` find `spec.dnsNames` and replace the line that begins `- <INSERT SITE HOSTNAME HERE>` with `- hostname of the site`.

4. Use the following steps to allow `ingress-ca` secrets to be the same on both data centers.
   a. On DC1 apply the file `custom-certs-external-dc1.yaml`:

   ```
   kubectl -n <namespace> apply -f custom-certs-external-dc1.yaml
   ```

   b. Validate that the command succeeded:

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

   c. Export `ingress-ca` secret as a yaml from DC1:

   ```
   kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
   ```

   d. Edit the `ingress-ca.yaml` file to remove all labels, annotations, `creationTimestamp`, `resourceVersion`, `uid`, and `selfLink`.
   e. Copy the `ingress-ca.yaml` from DC1 to DC2 and apply that file on DC2:

   ```
   kubectl -n <namespace> apply -f ingress-ca.yaml
   ```

   f. On DC2 apply the file `custom-certs-external-dc2.yaml`:

   ```
   kubectl -n <namespace> apply -f custom-certs-external-dc2.yaml
   ```

   g. Use the following commands to test that they are the same, on DC1 run:

   ```
   kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
   /tmp/ingress.pem.dc1
   ```

   h. On DC2 run:

   ```
   kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
   /tmp/ingress.pem.dc2
   ```

   i. To see the differences run:

   ```
   diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
   ```

   The files should be the same.
   j. To ensure that the certificates are working correctly and that they are using the `ingress-ca` secret. First, get the `portal-admin-client crt` file:

   ```
   kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
   /tmp/admin-client.crt
   ```

   k. Test that it is working by using OpenSSL:

   ```
   openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
   ```

   If it is working, you should see:

   ```
   /tmp/admin-client.crt: OK
   ```

5. Update the API Connect subsystem CR .yaml files to use the custom certificates. To do this, continue with Configuring custom certificates before installation.

Configuring custom certificates before installation

Configure custom certificates for your API Connect deployment before you begin the installation procedures.

About this task

To deploy API Connect with custom certificates, complete the following steps to configure your custom certificates before applying any subsystem installation CRs to the cluster.

Procedure

1. Determine which certificates you want to customize.
   For a list of certificates that can be customized, see Custom certificates reference.

   You can customize the following types of certificates:

   - Endpoint (External Frontend/Ingress) certificates:
     These are the certificates for external, public-facing endpoints where users interact with API Connect. You can customize none, some, or all of the endpoint certificates.

     If you initially deploy APIC using default certificates for endpoints, you can convert them to custom certificates later as explained in Converting to custom front-end/ingress certificates after deployment.

   - Internal (CA, Client, Server, and Common Subsystem Communication) cerificates:
     These are the certificates used within API Connect, so that subsystems can communicate. If you want to customize any of the internal certificates, then you must customize all of them; however you don't have to create them all yourself. Any custom certs that are not specifically named in your deployment CR will be generated automatically during installation.

     If you initially deploy API Connect using default certificates for internal certificates, you cannot convert them to custom certificates later.

2. Prepare your custom certificates.
   If you don't have a set of custom certificates already, you can use Kubernetes Cert-manager to generate them as explained in Generating custom certificates.

   When preparing your custom certificates, remember that certain pairs of certificates must by signed by the same Certificate Authority, as noted in Custom certificates reference.

   If you're installing a two data center disaster recovery configuration, and are using custom certificates, then the subject names of any client certificates must be the same in both data centers, as noted in Custom certificates reference. For example, both data centers subject name could be **`CN=portal-admin-client`**, or they could both be **`CN=ptl-adm-client, O=cert-manager`**, but they must be identical.

3. If needed, create a Kubernetes secret for each custom certificate.
   Each custom certificate that you use must be captured in a TLS secret. To create a secret, use the following commnad:

   ```
   kubectl create secret generic my-tls-secret --from-file=tls.crt=<path_to_cert_file> --from-file=tls.key=<path_to_key_file>
   --from-file=ca.crt=<path_to_cacert_file>
   ```

   When you create each secret, it's helpful to reference its purpose or its corresponding certificate in the secret name, to ensure you specify the correct secret for each custom certificate.

4. Edit the ingress-issuer-v1.yaml file and delete the default definitions for the certificates that you will customize.
   The ingress-issuer-v1.yaml file is provided as part of the installation **`helper_files`** archive.

   This file is used for generating the default certificates. To use a custom certificate, you must delete the default definition to ensure that API Connect uses your certificate instead of the default certificate. If you customize all of the default certificates and delete those definitions from the file, then you don't need to use the file during installation.

5. Customize endpoint (External Frontend/Ingress) certificates in the corresponding subsystem CRs.
   For each custom endpoint certificate, delete the annotation that specifies the default issuer, and replace the secret name with your own custom secret name.

   a. Open the subsystem CR for editing and locate the endpoint definition.
      For example, the following snippet shows the definition for the **`cloudManagerEndpoint`** in the management CR:

      ```
      cloudManagerEndpoint:
        annotations:
          cert-manager.io/issuer: ingress-issuer
        hosts:
        - name: admin.example.com
          secretName: cm-endpoint
      ```

   b. Delete the annotation for the **`cert-manager.io/issuer`**.
      Delete the field name as well as the value; you can leave the **`annotations:`** label in place.

   c. Replace the default secret name with the name of the Kubernetes secret that you created for that endpoint's custom certificate.
      In the following example, **`cm-endpoint`** is the default secret name, which must be replaced. Make sure to reference the correct secret for the custom certificate.

      ```
      cloudManagerEndpoint:
        annotations:
        hosts:
        - name: admin.example.com
          secretName: my_cm-endpoint
      ```

6. Customize internal (CA, Client, Server, and Common Subsystem Communication) certificates.
   a. Open the subsystem CR for editing.
   b. Set the **`microServiceSecurity`** setting to **`custom`**.
      By default, the setting is configured for **`certManager`** as shown in the following example:

      ```
      microServiceSecurity: certManager
      certManagerIssuer:
          name: selfsigning-issuer
          kind: Issuer
      ```

   c. Delete (or comment-out) the **`certManagerIssuer`** block.
      In the following example, **`microServiceSecurity`** is set to **`custom`** and the **`certManagerIssuer`** block is commented out:

      ```
      microServiceSecurity: custom
      #certManagerIssuer:
      #    name: selfsigning-issuer
      #    kind: Issuer
      ```

7. Add the **`customCertificates`** block and list the secret name for each internal custom certificate that you created for the subsystem.
   Each subsystem supports a different set of Common Subsystem Communication and External Frontend/Ingress certificates. You do not have to provide a custom version of every certificate supported by the subsystem. Just list the custom certificates that you created; any certificates that you omit from the list will be generated automatically during installation.

   For example, your **`customCertificates`** list for the management subsystem might look like the following example:

   ```
   customCertificates:
     - name: caCertificate
       secretName: ingress-ca
     - name: clientCertificate
       secretName: management-client
     - name: serverCertificate
       secretName: management-server
     - name: dbServerCertificate
       secretName: db-server-certificate
     - name: NATSTLSCertificate
       secretName: management-natscluster-mgmt
     - name: dbClientPostgres
       secretName: db-client-postgres
     - name: dbClientReplicator
   ```

```
    secretName: db-client-replication
  - name: dbClientApicuser
    secretName: db-client-apicuser
```

Note: On v10.0.6, the certificates are:

```
customCertificates:
  - name: caCertificate
    secretName: my_ingress-ca
  - name: clientCertificate
    secretName: my_management-client
  - name: serverCertificate
    secretName: my_management-server
  - name: dbServerCertificate
    secretName: my_db-server-certificate
  - name: pgBouncerServerCertificate
    secretName: my_pg-bouncer-server-certificate
  - name: PGOTLSCertificate
    secretName: my_pgo.tls
  - name: NATSTLSCertificate
    secretName: my_management-natscluster-mgmt
  - name: dbClientPostgres
    secretName: my_db-client-postgres
  - name: dbClientReplicator
    secretName: my_db-client-replicator
  - name: dbClientPgbouncer
    secretName: my_db-client-pgbouncer
  - name: dbClientApicuser
    secretName: my_db-client-apicuser
  - name: dbClientPrimaryuser
    secretName: my_db-client-primaryuser
```

What to do next

Deploy API Connect:

- If you did not deploy the API Connect operator yet, proceed to Deploying operators in a single-namespace API Connect cluster or Deploying operators in a multi-namespace API Connect cluster.
- If you already deployed the operator, proceed to Installing the API Connect subsystems.

Converting to custom front-end/ingress certificates after deployment

Convert a front-end certificate generated by cert-manager to a custom certificate for an existing IBM® API Connect deployment.

About this task

If you deploy IBM API Connect using certificates that were generated by cert-manager, you can later convert front-end (ingress) certificates to use custom certificates instead. For a list of the certificates that are generated and managed by cert-manager, see Custom certificates reference.

Restriction: This task applies only to front-end certificates and is not supported for the "Common Subsystem Communication" and "Internal" type certificates.

Procedure

1. Create the secrets you want to use with the `kubectl create secret` command.
2. Edit the subsystem CR, and make the following changes for the endpoint secret you want to change:
   - In the `annotations` section, remove the following line:

     ```
     cert-manager.io/issuer: ingress-issuer
     ```

   - In the `hosts` section, update the `secretName` to the new secret that you created in step 1.

   The following example shows where the updates should be made:

```
spec:
  <endpoint>
    annotations:
      cert-manager.io/issuer: ingress-issuer  REMOVE THIS LINE
    hosts:
    - name: <api endpoint hostname>
      secretName:  CHANGE THIS TO THE NEW SECRET
```

Webhooks deployed and managed by the API Connect Operator

The API Connect Operator deploys and manages various Kubernetes API webhooks in the cluster to assist in management of its custom resources.

The webhooks included admission controllers and conversion webhooks.

Defaulting webhook

- Purpose

The defaulting webhook is a type of Mutating Admission Webhook which runs against Custom Resources (CRs) at **CREATE** or **UPDATE** time to populate default values in the CR's **spec**.

- Creation
  Currently the defaulting webhook runtime is part of the API Connect Operator runtime, that is it exists within the API Connect Operator pod and container. When the operator boots, it creates **MutatingWebhookConfiguration** resources for each CRD (listed below). This **MutatingWebhookConfiguration** is a cluster-scope resource, but the configuration within will be specific to the namespace and operator instance which created it. If more than one instance of the API Connect Operator is deployed across a cluster (in different namespaces) there will be a set of webhook configurations for each API Connect Operator instance and namespace.

  The name for the **MutatingWebhookConfiguration** resource for each CRD will take the following form:

  ```
  <namespace>.<crd-name>.defaulter[.<component>].apiconnect.ibm.com
  ```

  The following CustomResourceDefinitions utilize a defaulting webhook:

  ```
  AnalyticsCluster
  ApiconnectCluster
  GatewayCluster
  ManagementCluster
  ManagementDBUpgrade
  ManagementRestore
  PortalCluster
  ```

- Lifecycle
  While the defaulting webhook runtime itself lives within the API Connect Operator pod, the **MutatingWebhookConfiguration** is a cluster-scope resource and is stand-alone. When it is created, an **ownerReference** is set on it that references the API Connect Operator's **ClusterRole**. Thus, the lifecycle of the **MutatingWebhookConfiguration** resources will be linked with the **ClusterRole**, which is linked with the installation of the operator itself.

  When the operator is uninstalled, the **MutatingWebhookConfiguration** resources for that operator instance are also removed.

- Example
  The below output shows the defaulting webhook configurations created for an operator deployed in the **apiconnect-operator** namespace. One can be seen for each CRD.

  ```
  $ oc get mutatingwebhookconfiguration
  NAME                                                                  WEBHOOKS   AGE
  apiconnect-operator.analyticsclusters.defaulter.analytics.apiconnect.ibm.com       1          4d4h
  apiconnect-operator.apiconnectclusters.defaulter.apiconnect.ibm.com                1          4d4h
  apiconnect-operator.gatewayclusters.defaulter.gateway.apiconnect.ibm.com           1          4d4h
  apiconnect-operator.managementclusters.defaulter.management.apiconnect.ibm.com    1          4d4h
  apiconnect-operator.managementdbupgrades.defaulter.management.apiconnect.ibm.com  1          4d4h
  apiconnect-operator.managementrestores.defaulter.management.apiconnect.ibm.com    1          4d4h
  apiconnect-operator.portalclusters.defaulter.portal.apiconnect.ibm.com             1          4d4h
  ```

Validating webhook

- Purpose
  The validating webhook is a type of ValidatingAdmissionWebhook which runs against Custom Resources (CRs) at **CREATE** or **UPDATE** time to validate the CR contains a configuration which will yield an operable cluster. The CustomResourceDefinitions use OpenAPI schema validation for basic property checks (such as integer bounds and non-empty strings), but more complex validation of the configuration is done via the webhook.

- Creation
  Currently the validating webhook runtime is part of the API Connect Operator runtime, i.e. it exists within the API Connect Operator pod and container. When the operator boots, it creates **ValidatingWebhookConfiguration** resources for each CRD (listed below). This **ValidatingWebhookConfiguration** is a cluster-scope resource, but the configuration within will be specific to the namespace and operator instance which created it. If more than one instance of the API Connect Operator is deployed across a cluster (in different namespaces) there will be a set of webhook configurations for each API Connect Operator instance and namespace.

  The name for the **ValidatingWebhookConfiguration** resource for each CRD will take the following form:

  ```
  <namespace>.<crd-name>.validator[.<component>].apiconnect.ibm.com
  ```

  The following CustomResourceDefinitions utilize a validating webhook:

  ```
  AnalyticsCluster
  ApiconnectCluster
  GatewayCluster
  ManagementCluster
  ManagementDBUpgrade
  ManagementRestore
  PortalCluster
  ```

- Lifecycle
  While the validating webhook runtime itself lives within the API Connect Operator pod, the **ValidatingWebhookConfiguration** is a cluster-scope resource and is stand-alone. When it is created, an **ownerReference** is set on it that references the API Connect Operator's **ClusterRole**. Thus, the lifecycle of the **ValidatingWebhookConfiguration** resources will be linked with the **ClusterRole**, which is linked with the installation of the operator itself.

  When the operator is uninstalled, the **ValidatingWebhookConfiguration** resources for that operator instance are also removed.

- Example
  The below output shows the validating webhook configurations created for an operator deployed in the **apiconnect-operator** namespace. One can be seen for each CRD.

  ```
  validatingwebhookconfiguration
  NAME                                                                  WEBHOOKS   AGE
  apiconnect-operator.analyticsclusters.validator.analytics.apiconnect.ibm.com       1          4d4h
  apiconnect-operator.apiconnectclusters.validator.apiconnect.ibm.com                1          4d4h
  ```

```
apiconnect-operator.gatewayclusters.validator.gateway.apiconnect.ibm.com          1          4d4h
apiconnect-operator.managementclusters.validator.management.apiconnect.ibm.com    1          4d4h
apiconnect-operator.managementdbupgrades.validator.management.apiconnect.ibm.com  1          4d4h
apiconnect-operator.managementrestores.validator.management.apiconnect.ibm.com    1          4d4h
apiconnect-operator.portalclusters.validator.portal.apiconnect.ibm.com            1          4d4h
```

Administering API Connect in your container environment

How to manage your API Connect subsystems in Kubernetes and OpenShift environments.

On all container platforms, your API Connect subsystems can be entirely managed from the command line, by using the Kubernetes **kubectl** command. On OpenShift, you can use the equivalent **oc** command instead of **kubectl**. However, since the **kubectl** command is recognized in OpenShift, the **kubectl** command is used in most example commands throughout this documentation.

On OpenShift, some administration operations can be done through the OpenShift console, and on Cloud Pak for Integration also through the Platform UI.

- **Namespaces in your container environment**
  How to administer API Connect in a namespace-scoped container environment.
- **Custom Resources and Custom Resource Definitions**
  Information about the Custom Resources (CRs) that make up API Connect.
- **Managing your API Connect CRs**
  How to update the API Connect CRs in your deployment.

Namespaces in your container environment

How to administer API Connect in a namespace-scoped container environment.

Namespaces are used in Kubernetes to allow separate groups of resources to run in the same Kubernetes cluster without interfering with each other. For example, you can have two Kubernetes pods with an identical name if they are in different namespaces.

When you install API Connect, you can use an existing namespace or create a new namespace for your API Connect installation to run in.

Cluster-scoped resources are Kubernetes resources that operate across multiple namespaces. The API Connect operator is installed at cluster-scope, and can manage multiple API Connect installations that are installed in different namespaces.

Whenever you run the **kubectl** command, you must specify which namespace the command is to run in. For example:

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

where `-n <namespace>` defines the namespace to run the command in. If you specify the wrong namespace, the **kubectl** command fails:

```
kubectl -n wrong-ns get pod mypod
Error from server (NotFound): namespaces "wrong-ns" not found
```

Kubernetes and OpenShift have a default namespace that is used if no namespace is specified in **kubectl** commands. It is not recommended to install API Connect in the default namespace. Always create a dedicated namespace for each API Connect installation.

Throughout this document, **kubectl** commands are provided to run the various installation and maintenance operations that are used to manage your API Connect installation. For brevity, the `-n <namespace>` is not included in the example commands where the namespace to specify is clear from the context.

If you need to run a sequence of `kubectl` commands in the same namespace, you can set the default namespace so that you do not need to include `-n <namespace>` in the commands. To set the default namespace, run the following command:

- Kubernetes:

  ```
  kubectl config set-context --current --namespace=<namespace>
  ```

- OpenShift:

  ```
  oc project <namespace>
  ```

Custom Resources and Custom Resource Definitions

Information about the Custom Resources (CRs) that make up API Connect.

Custom Resources (CRs) are sets of Kubernetes resources (such as pods) that you can interact with using the **kubectl** command. An API Connect installation has a distinct CR for each API Connect subsystem: management, portal, gateway, and analytics.

Custom Resource Definitions (CRDs) are the Kubernetes resource that is used to define CRs. API Connect has a CRD for each subsystem:

- ManagementCluster
- PortalCluster
- GatewayCluster
- AnalyticsCluster

API Connect on Cloud Pak for Integration or on OpenShift with the top-level CR, has an extra CRD for the top-level CR:

- APIConnectCluster

Each API Connect subsystem CR defines the system-level properties for its subsystem such as endpoints, secrets, backup settings, and storage allocations. The properties are defined in YAML format. The syntax to view the properties of an API Connect CR are:

```
kubectl -n <namespace> get <API Connect CRD name> <API Connect CR name> -o yaml
```

Example output:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  annotations:
    ...
  creationTimestamp: "2023-12-06T17:37:28Z"
    ...
  name: management
  namespace: apic
  ...
  uid: 9e67df02-d044-4a5b-8fb5-93391722ae97
spec:
  adminUser:
    secretName: management-admin-pass
  ...
  analytics:
    client: {}
    ingestion:
      secretName: management-a7s-ing-client
  apiManagerEndpoint:
    annotations:
      cert-manager.io/issuer: management-ingress-issuer
      haproxy.router.openshift.io/timeout: 240s
    hosts:
    - name: management-api-manager-apic.apps.ibm.com
      secretName: management-api-manager
    ...
  databaseBackup:
    ...
  databaseVolumeClaimTemplate:
    storageClassName: rook-ceph-block
    ...
  version: 10.0.x.y
  ...
status:
  conditions:
  - lastTransitionTime: "2023-12-06T17:47:47Z"
    message: All services ready
    reason: na
    status: "True"
    type: Ready
...
```

To refer to a specific property in the CR, dot notation is used, so the property:

```
metadata:
  annotations:
```

is referred to as `metadata.annotations`. In the example output shown previously, some properties of interest are highlighted in bold:

- `kind: ManagementCluster`. The `kind` property refers to the CRD that defines the CR.
- `name: management`. The name of the CR itself. When you want to run a kubectl operation on the CR, you specify the CR name in the command.
- `namespace: apic`. The name of the namespace the CR exists in.
- `spec:` The `spec` section contains the properties of your API Connect subsystem, such as URLs for the endpoints and scheduled database backup settings.
- `spec.apiManagerEndpoint.hosts.name` is where you define the URL of the API Manager UI.
- `status:` The `status` section shows information about the current state of the subsystem, including any errors or warnings.

You can have multiple portal, analytics, and gateway CRs in the same namespace, each registered with the management subsystem as separate services. You cannot have multiple management CRs in the same namespace. To list all the installed CRs for a portal, analytics, or gateway CRD, run:

```
kubectl get <CRD name>
```

where *<CRD name>* is `PortalCluster`, `GatewayCluster`, or `AnalyticsCluster`. Example output:

```
kubectl get PortalCluster
```

```
NAME        READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE                                                AGE
ptl_1       3/3     Running   10.0.x.y   10.0.x.y-3968        Ready for API Cloud Manager service registration   6d22h
ptl_2       3/3     Running   10.0.x.y   10.0.x.y-3968        Ready for API Cloud Manager service registration   9d12h
```

The output shows two portal CRs. To get just information on one of the CRs, specify that CR name in the **kubectl get** command:

```
kubectl get PortalCluster ptl_2
```

```
NAME        READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE                                                AGE
ptl_2       3/3     Running   10.0.x.y   10.0.x.y-3968        Ready for API Cloud Manager service registration   9d12h
```

Tip: Instead of typing the full CRD names `ManagementCluster`, `PortalCluster`, `GatewayCluster`, or `AnalyticsCluster`, you can use the following shorter alternatives:

- `mgmt` for `ManagementCluster`
- `ptl` for `PortalCluster`
- `gw` for `GatewayCluster`
- `a7s` for `AnalyticsCluster`

To see all the API Connect CRs in a namespace, use the `kubectl get apic` command:

```
kubectl get apic -n <namespace>
NAME
                                                    READY   STATUS    VERSION    RECONCILED VERSION   AGE
analyticscluster.analytics.apiconnect.ibm.com/test-a7s   n/n     Running   10.0.x.y   10.0.x.y-2248        5d

NAME                                                READY   STATUS    VERSION    RECONCILED VERSION   AGE
gatewaycluster.gateway.apiconnect.ibm.com/test-gw   n/n     Running   10.0.x.y   10.0.x.y-2248        5d

NAME                                                   READY   STATUS    VERSION    RECONCILED VERSION   AGE
managementcluster.management.apiconnect.ibm.com/test-mgmt   n/n   Running   10.0.x.y   10.0.x.y-2248        5d1h

NAME                                                 READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE
AGE
portalcluster.portal.apiconnect.ibm.com/test-ptl   n/n     Running   10.0.x.y   10.0.x.y-2248        Ready to create sites   5d
```

Managing your API Connect CRs

How to update the API Connect CRs in your deployment.

Updating an API Connect CR requires editing the CR and adding, updating, or deleting individual properties or entire sections of the CR. Two methods exist to edit a CR:

- Edit the CR manually with the **kubectl** edit command. The **kubectl** edit command opens the CR in an editor, where standard **vi** operations can be used to update the CR:

  ```
  kubectl edit ptl <portal cr name>
  ```

  Make the changes that you want in the CR, for example update the value of `spec.databaseBackup.schedule`, then save and exit.
  Note: You can omit the CR name in the edit command. To open all portal CRs in the namespace for editing, run:

  ```
  kubectl edit ptl
  ```

  For brevity, the CR name is often omitted where **kubectl** edit commands are provided in this documentation.

- Use the **kubectl** `patch` operation to update the CR in a single command, for example:

  ```
  kubectl patch managementcluster <mgmt cr name> --type json -p='[{"op": "remove", "path": "/spec/multiSiteHA"}]'
  ```

  In this example, the patch operation removes the entire `spec.multiSiteHA` section from the management CR.

  Patch operations that require updating multiple properties in the same CR can be done by creating a YAML file that contains all the properties to change, for example:

  ```
  kubectl patch mgmt <mgmt cr name> --type merge --patch "$(cat <YAML file>)"
  ```

  where *<YAML file>* is the file that contains the properties to update.
  Note: With `patch` operations, you must always specify the CR name, even if you have only one CR of the CRD type in the namespace.

Both methods for updating CRs are used throughout this documentation.

Installing API Connect

Use these instructions to install a deployment of API Connect on Kubernetes, OpenShift, and IBM Cloud Pak for Integration.

To deploy API Connect, complete the tasks in the order below.

- **Deployment requirements**
  Describes the system requirements for deploying into the Kubernetes, OpenShift, and Cloud Pak for Integration runtime environments.
- **Deployment procedures**
  You can use these procedures to deploy API Connect on either native Kubernetes or OpenShift.

Deployment requirements

Describes the system requirements for deploying into the Kubernetes, OpenShift, and Cloud Pak for Integration runtime environments.

Before you begin

Note: This article refers to third-party software that IBM does not control. As such, the software might change and this information might become outdated.
Kubernetes is a platform for automated deployment, scaling, and operation of application containers across clusters of hosts, providing container-centric infrastructure.
For more information, see https://kubernetes.io.

These instructions assume you have a working Kubernetes environment and understand how to manage Kubernetes.

About this task

Review all the requirements listed in the following steps before starting your API Connect deployment.

Note: If you plan to install the Automated API behavior testing application, review the *Before you begin* section in Installing the Automated API behavior testing application because the requirements might have an impact on your deployment choices for API Connect.

Procedure

1. Verify that you have supported software.
   - Refer to the IBM Software Product Compatibility Reports site for detailed requirements for operating systems, supporting software, and other prerequisites. See Detailed system requirements for a specific product
   - For new installations, the API Connect operators and operands must be from the same version of API Connect. This requirement applies to native Kubernetes and OpenShift installations.
     - Note that the API Connect operator version *number* does not match the API Connect release version *number*. You can view a mapping of API Connect release number to API Connect operator version numbers in Deploying operators and cert-manager.
     - For API Connect operands (Management, Portal, Analytics, and Gateway subsystems), the version number matches the API Connect release version number.

2. Verify that your Kubernetes environment is running and meets the configuration requirements for API Connect:
   - When using IBM Cloud Kubernetes Service, ingress-nginx is required.
   - Specify settings for the ingress controller for a Kubernetes runtime environment. See Kubernetes ingress controller prerequisites.
   - The timezone for API Connect pods is set to UTC. Do not change the timezone for the pods.
   - To ensure the API Connect services have time to start, we recommend increasing the *proxy-read-timeout* and *proxy-send-timeout* values in the *config.map* used to configure the kubernetes/ingress-nginx ingress controller. The following settings should be increased:
     - proxy-read-timeout: "240"
     - proxy-send-timeout: "240"

     240 seconds (4 minutes) is a recommended minimum; actual value will vary depending upon your environment. If there is a load balancer in front of the worker node(s), then the load balancer configuration may also need to have extended timeouts.
   - Storage type:
     When selecting storage type, review the following considerations:

     - Management subsystem requires block storage.
     - Developer Portal subsystem supports block storage or local volume storage.
     - Analytics subsystem supports block storage or local volume storage.

     Supported storage types
     - Block storage
       Select a block storage format of your choosing in order for API Connect components to be supported. Supported block storage types include, but are not limited to:

       - Rook Ceph
         Note: cephrbd (RADOS Block Device) is recommended over cephfs file system

       - OpenShift Container Storage
       - Portworx
       - IBM Cloud Block Storage
       - AWS: the gp2, gp3, io1, and io2 types of Elastic Block Storage (EBS) are supported.
       - IBM Spectrum Scale with Block Storage
       - Azure Premium Storage
       - GCE Persistent Disk

       See also the **API Connect v10 WhitePaper 1.5** for information on limitations observed with other block storage solutions:
       https://community.ibm.com/community/user/integration/viewdocument/api-connect-deployoment-whitepaper-v?CommunityKey=2106cca0-a9f9-45c6-9b28-01a28f4ce947&tab=librarydocuments&LibraryFolderKey=&DefaultView=folder

     - Local volume storage

       Management subsystem and Developer Portal subsystem
       All subsystems will run on local volume storage but it is not recommended for the Management subsystem and Developer Portal subsystem because it has severe performance impacts for high availability deployments, and for pod allocation.
       Note also that local volume storage is not available in most cloud environments, and might incur limitations in disaster recovery scenarios, since it is tied to the node on which the pod is run. This means that use of LVS might be adequate for deployments with a one replicaprofile, but inadequate for deployments with a three replica profile.

       Analytics subsystem
       For Analytics subsystems, transactions per second (TPS) is three times faster with local volume storage than with Ceph block storage. In a production scenario, you will have multiple analytics-storage (OpenSearch) nodes forming a storage cluster. OpenSearch will manage data replication across its cluster and ensure high availability, thus mitigating possible local volume storage drawbacks with disaster recovery.

     Non-supported storage
     - NFS
       API Connect v10 cannot be deployed on NFS.

     - GlusterFS
       GlusterFS is not recommended as a storage option due to severe performance degradation and in some cases, loss of data. There are known GlusterFS issues that affect OpenSearch (which is used in the API Connect analytics service), the Developer Portal and the Management subsystem

   - Storage class configuration:
     In your Kubernetes cluster configuration, you must create storage classes with the setting:

     `volumeBindingMode: WaitForFirstConsumer`

     This setting is needed to avoid volume node **`affinity`** conflicts, such as those caused by **`requiredPodAntiAffinity`** restrictions. If **`WaitForFirstConsumer`** is the binding mode, PVCs (PersistentVolumeClaims) are only bound once the pod is actually created.

     By contrast, when the binding mode is immediate (**`volumeBindingMode: Immediate`**), the PVCs are bound to PVs (PersistentVolumes) as soon as the PV is created.

- Example:
  On v10.0.6 the Postgres operator creates the PVC first and then deploys the deployments. In immediate mode, PVCs are bound to PVs as soon as the PV is created. The **pgbackrest** pod starts successfully, but because **requiredPodAntiAffinity** restrictions are set, and the **postgres** pod is **NOT ALLOWED** to schedule on the same node as **pgbackrest** pod, Kubernetes cannot schedule the pod in node **eu-west-1c**. Also, because the volume attached to the pod is at **eu-west-1c**, Kubernetes cannot move the pod to any other node, thus causing **volume node affinity** conflict.

  On v10.0.7, there is no **pgbackrest** pod.

- Example:
  When deploying to a multi-zone cluster, you must use StorageClass with **WaitForFirstConsumer** because when binding mode is immediate, successful deployment is indeterminate because of interactions between pod/node affinity and the Kubernetes provisioning of the PV on the right node where the pod is being provisioned.

For more information, see "Volume Binding Mode" on https://kubernetes.io/docs/concepts/storage/storage-classes/.

Note: During deployment, you are required to specify the name of the storage class you created. You can either use the setting **storageClassName: $STORAGE_CLASS** in the applicable CR (such as apiconnect top-level CR or an individual subsystem CR), or you can use a configuration UI such as Platform Navigator UI or OpenShift Container Platform (OCP) UI.

- When the Analytics service is configured to store data, (that is, it is not configured for ingestion-only), the service depends on OpenSearch, which requires map counts higher than the operating system defaults. See: Installing the Analytics subsystem.
  If you are deploying on OpenShift, the map count is preconfigured (**vm.max_map_count = 1027580**) with a value that is sufficiently large for OpenSearch so you do not need to modify this setting.

- Ensure that your Kubernetes deployment addresses known security vulnerabilities with SSH to prevent the use of weak SSH Message Authentication Code (MAC) and Key exchange algorithms (KexAlgorithms) with TCP over port 22. Use of weak algorithms can allow an attacker to recover the plain text message from the encrypted text. If you have not yet addressed the security vulnerabilities with weak MACs and KexAlgorithms, update your configuration manually by adding the following lines to /etc/ssh/sshd_config:

  ```
  KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256
  MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
  ```

- IBM API Connect requires a number of container registry repositories to be added to your container registry service if automatic repository creation is not supported. Follow the procedure in Step 4 of Obtaining product files.

- Logging:
  - Configure a remote server for logging. See Logging. Pods are terminated during an upgrade, and the logs will be lost if not stored remotely.
  - Configure the log rotation by setting the **max-size** and **max-files** values to a setting appropriate for your deployment. In a Kubernetes deployment, insufficient values for max-size and max-files can cause logs to grow too large and eventually force pods to restart.
    For more information, visit the Docker documentation and search for "JSON File logging driver".

3. Ensure your deployment meets the DNS requirements for API Connect domain names, host names, and endpoints.
   You specify endpoints when you edit the custom resource templates for each subsystem, and when you use the Cloud Manager UI to configure the services for your cloud. The domain names you use for the endpoints must be recognized by your DNS configuration. DNS entries must meet these requirements:

   Important: For API Connect on native Kubernetes and OpenShift, do not use coredns 1.8.1 through 1.8.3. These releases have a defect that can prevent pods for Developer Portal from starting. See https://coredns.io/.

   - Endpoints cannot contain the underscore character "_" or uppercase characters.
   - Host names that are used to create endpoints cannot contain the underscore character "_" or uppercase letters.
   - DNS must be configured with domain names that correspond to the endpoints configured at installation time for each subsystem.
   - Host names and DNS entries may not be changed on a cluster after the initial installation.
   - Kubernetes ingress limits the character set for DNS names to not support the underscore character "_". This means you cannot specify underscores in domain names that are used as endpoints. For example, **my_domain.abc.com** and **my.domain_abc.com** are not supported for **<xxx>.<hostname>.<domainname>**, and will cause an error:

     ```
     Invalid value: "my_domain.abc.com": a DNS-1123 subdomain must consist of lowercase alphanumeric characters, '-' or '.',
     and must start and end with an alphanumeric character (e.g. 'example.com', regex used for validation is
      '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')
     ```

   - We recommend that you keep a record of the DNS entries and endpoint names, as you will need to map the DNS entries and use the same endpoints when restoring a backup of the management database. Note that endpoints are FQDNs, entered in all lowercase. For an overview on endpoints, review Deployment overview for endpoints and certificates
   - Specifying domain names when deploying CRs (custom resources) for each subsystem:
     The custom resource templates provide endpoint settings that consist of a default endpoint prefix and a variable **$STACK_HOST** that you set to specify your domain name, as part of Installing the API Connect subsystems.

   Table 1. Endpoints with domain names that must be configured in DNS

Default endpoint name	Description
admin.$STACK_HOST	Cloud Manager URL endpoint for the Management subsystem
manager.$STACK_HOST	API Manager URL endpoint for the Management subsystem
api.$STACK_HOST	Platform REST API Endpoint for admin and provider APIs
consumer.$STACK_HOST	Platform REST API Endpoint for consumer APIs for the Management subsystem.
rgw.$STACK_HOST	Gateway Endpoint for API invocation for DataPower API Gateway
rgwd.$STACK_HOST	Gateway Management Endpoint for DataPower API Gateway
gw.$STACK_HOST	Gateway Endpoint for API invocation for DataPower Multi-Protocol Gateway (v5-compatible)
gwd.$STACK_HOST	Gateway Management Endpoint for DataPower Multi-Protocol Gateway (v5-compatible)
ai.$STACK_HOST	Analytics ingestion endpoint for the Analytics subsystem
api.portal.$STACK_HOST	Developer Portal admin endpoint
portal.$STACK_HOST	Developer Portal UI endpoint

   - Specifying endpoints using the Cloud Manager interface

After initial deployment of the Kubernetes custom resource for each subsystem, you will also enter the endpoints in the Cloud Manager user interface when defining each service as part of Defining your topology.

4. The Cloud Provider for the target Kubernetes platform must provide Ingress Controller support for SSL Passthrough.
   For example, if the Cloud Provider uses or suggests the use of the NGINX Community Ingress, then the **`--enable-ssl-passthrough`** flag would need to be provided at runtime, as described at https://kubernetes.github.io/ingress-nginx/user-guide/tls/#ssl-passthrough. For IBM Cloud Kubernetes Service, see https://cloud.ibm.com/docs/containers?topic=containers-ingress-user_managed..

5. Determine your strategy for using certificates and cert-manager with API Connect.
   If you haven't already, review the topics in the section Certificates in a Kubernetes environment.

   Consider the following:

   - Will you use a cert-manager (recommended) or manually manage your certificates?
     API Connect provides a cert-manager and default certificates. If you use these, you can follow default configuration steps. See Deploying operators and cert-manager.

   - Will you use custom certificates rather than default certificates? See Custom certificates on Kubernetes.

6. If you plan to deploy API Connect on Kubernetes with DataPower Gateway in a *non-Kubernetes* environment, such as a DataPower appliance, review the API Connect requirements for operation with DataPower Gateway:
   - When deploying with a DataPower appliance, configure a Gateway Service Management Endpoint and API invocation Endpoint in DataPower. See *Configuring the API Connect Gateway Service* in the appropriate version of the DataPower documentation.
   - Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.
     Note:
     You can use any combination of API Connect CD releases (10.0.6.x) with DataPower CD releases (DataPower Gateway 10.5.1.x or DataPower API Gateway 10.5.1.x).

     Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

7. If you plan to deploy the API Connect management subsystem and the DataPower Gateway subsystem on a Kubernetes deployment, managed by the DataPower and API Connect Operators, you can use any combination of API Connect 10.0.5.x with DataPower Gateway 10.5.0.x or DataPower API Gateway 10.5.0.x.
8. Observe namespace restrictions:
   - OpenShift
     Only one top level CR (APIConnectCluster) can be deployed in each namespace.

   - Cloud Pak for Integration
     In the Platform Navigator UI, only one APIConnect capability can be deployed in each namespace.

   - Native Kubernetes
     Only one instance of a subsystem can be deployed in each namespace. Deploying multiple similar subsystems in a single namespace is not supported. For example, there is no support for deploying two management subsystems, **`mgmt1`**, **`mgmt2`** in a single namespace where apiconnect operator is installed.

9. The following kernel or Kernetes node limits for the Developer Portal subsystem must be set correctly:
   - Ensure that your kernel or Kubernetes node has the value of its **`inotify`** watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a **`non-ready`** state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of **`inotify`** watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take **`inotify`** watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.
   - Ensure that your kernel or Kubernetes node has a value of **`nproc`** (maximum number of processes) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow all of the Portal processes to execute. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls, you might need as many as 125205. If the number is too low, you will see errors like **`"fork: retry: Resource temporarily unavailable"`** in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.
   - Ensure that your kernel or Kubernetes node has a value of **`nofiles`** (maximum number of open file descriptors) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow the Portal to open all of the files that it requires. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls and Portal web sites, you might need as many as 1048576. If the number is too low, you will see errors like **`"too many open files"`** in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.
10. Your Kubernetes environment must have access to an S3 object-store, for remote storage of API Connect subsystem database backups.

- **IBM API Connect Version 10 software product compatibility requirements**
  Ensure that you install the minimum API Connect operating system requirements. Use the IBM Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.
- **API Connect licenses**
  When you install a new version of API Connect, you must accept a license for the API Connect program that you purchased.

IBM API Connect Version 10 software product compatibility requirements

Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.

Generating a Software Product Compatibility Report

To generate an API Connect requirements report, complete the following steps:

1. Open the Detailed system requirements for a specific product page on the IBM Software Product Compatibility Reports site.
2. Search for the IBM API Connect product.
3. In the Search results list, select IBM API Connect.
4. From the Version list, select the required version.
5. Use the Filters to refine the contents of the requirements report.
6. Click Submit to generate your requirements report.

API Connect licenses

When you install a new version of API Connect, you must accept a license for the API Connect program that you purchased.

See also: API Connect licenses for the 10.0.5.x releases.
When you install or upgrade API Connect, the procedures include instructions for specifying your license. Be sure to specify a License ID from the version of API Connect that your installing, or are upgrading to.
Beginning with API Connect 10.0.6.0, the licenses in Table 1 will be used as the master mod-level license for all Fix Pack and security roll-up releases in this stream. There are no changes to license terms, only updates to open-source notices documented in the notes for each.

Table 1. API Connect 10.0.7 licenses

License ID	Program Name	URL
L-MMBZ-295QZQ	IBM API Connect Enterprise V10.0.7.0	https://ibm.biz/BdMLs3
L-GGQD-G7AYJD	IBM Cloud Pak for Integration - API Calls 2023.4.1	https://ibm.biz/BdMLiR
L-JWFC-XU2KXR	IBM API Connect Enterprise Add-on for IBM Products V10.0.7.0	https://ibm.biz/BdMLsb
L-LHBS-DGEM6J	IBM API Connect Professional V10.0.7.0	https://ibm.biz/BdMLsi

Table 2. API Connect 10.0.6 licenses

License ID	Program Name	URL
L-KZXM-S7SNCU	IBM API Connect Enterprise V10.0.6.0	https://ibm.biz/BdPJsn
L-NRHE-WFD7EY	IBM Cloud Pak for Integration - API Calls 2023.2.1	https://ibm.biz/BdPJse
L-TBPQ-X87AGB	IBM API Connect Enterprise Add-on for IBM Products V10.0.6.0	https://ibm.biz/BdPJsb
L-FNPP-MXMAH5	IBM API Connect Professional V10.0.6.0	https://ibm.biz/BdPJsp

Deployment procedures

You can use these procedures to deploy API Connect on either native Kubernetes or OpenShift.

- **Deploying on Kubernetes**
  Use the procedures in this section to install API Connect on native Kubernetes distributions.
- **Deploying on OpenShift and Cloud Pak for Integration**
  You can install API Connect in an OpenShift environment or as the API Management capability in IBM Cloud Pak for Integration.

Deploying on Kubernetes

Use the procedures in this section to install API Connect on native Kubernetes distributions.

- **Obtaining product files**
  Obtain the product files, upload the images to a Docker registry, and decompress the operators and templates.
- **Deploying operators and cert-manager**
  Deploy the Kubernetes operator files and install cert-manager.
- **Installing the API Connect subsystems**
  Deploy the API Connect subsystem custom resources.
- **Installing a two data center deployment on Kubernetes**
  Additional installation instructions for a two data center disaster recovery (2DCDR) deployment on Kubernetes.
- **Troubleshooting installation on Kubernetes**
  Review the following known issues and troubleshooting tips if you encounter a problem while installing API Connect on Kubernetes

Obtaining product files

Obtain the product files, upload the images to a Docker registry, and decompress the operators and templates.

Before you begin

- Ensure you have supported hardware and software. See IBM API Connect Version 10 software product compatibility requirements.
- Complete the Deployment requirements
- Install and run Docker on the local machine being used for API Connect installation. Log in to your image registry.

About this task

From the IBM Fix Central site, download the Docker image-tool file of the API Connect subsystems. Next, you will upload the image-tool file to your Docker local registry. If necessary, you can populate a remote container registry with repositories. Then you can push the images from the local registry to the remote registry.

You will also download the Kubernetes operators, API Connect Custom Resource (CR) templates, and Certificate Manager, for use during deployment configuration.

Note:

- If you plan to migrate from Version 5 to Version 10, review Migrating a version 5 deployment to version 10.0.x before you start the installation process.
- If you have API Connect installed and want to upgrade your release version, follow the upgrade instructions: Upgrading API Connect in a Kubernetes runtime environment

Procedure

1. Obtain the API Connect files:
   a. Go to the What's New in the latest version information page.
   b. Locate the Note: *You can access the latest files from <URL link>*. Select the *<URL link>* to go directly to the Announce page on Fix Central, where you can download files for the *latest version* of API Connect.

   The following files are used for initial deployment on native Kubernetes:

   IBM® API Connect *<version>* for Containers
   Docker images for all API Connect subsystems

   IBM® API Connect *<version>* Operator Release Files for Containers
   Kubernetes operators and API Connect Custom Resource (CR) templates

   IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
   Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:
   - IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
   - IBM® API Connect *<version>* Toolkit with Loopback for *<operating_system_type>*
   - IBM® API Connect *<version>* Toolkit Designer with Loopback for *<operating_system_type>*

   Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See Installing the toolkit.

   IBM® API Connect *<version>* Local Test Environment
   Optional test environment. See Testing an API with the Local Test Environment

   IBM® API Connect *<version>* Security Signature Bundle File
   Checksum files that you can use to verify the integrity of your downloads.

2. Optionally, you can verify the integrity of the downloaded files to ensure that they originated from IBM and are not modified. See Verifying the integrity of IBM product files.
3. Load the image-tool image in your Docker local registry. The image is contained in the IBM® API Connect *<version>* for Containers download file. For example:

   ```
   docker load < apiconnect-image-tool-<version>.tar.gz
   ```

   Ensure that the registry has sufficient disk space for the files.

4. If your Docker registry requires repositories to be created before images can be pushed, create the repositories for each of the images listed by the image tool. (If your Docker registry does not require creation of repositories, skip this step and go to Step 5.)
   a. Run the following command to get a list of the images from image-tool:

   ```
   docker run --rm apiconnect-image-tool-<version> version  --images
   ```

   b. From the output of each entry of the form `<image-name>:<image-tag>`, use your Docker registry repository creation command to create a repository for `<image-name>`.
   For example in the case of AWS ECR the command would be for each `<image-name>`:

   ```
   aws ecr create-repository --repository-name <image-name>
   ```

5. Upload the image:
   - If you do not need to authenticate with the docker registry, use:

     ```
     docker run --rm apiconnect-image-tool-<version> upload <registry-url>
     ```

   - Otherwise, if your docker registry accepts authentication with username and password arguments, use:

     ```
     docker run --rm apiconnect-image-tool-<version> upload <registry-url> --username <username> --password <password>
     ```

   - Otherwise, such as with IBM Container Registry, if you need the image-tool to use your local Docker credentials, first authenticate with your Docker registry, then upload images with the command:

     ```
     docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
     ```

     Note: The previous command does not work on macOS if Docker is configured to use the `osxkeychain` credential store. In this case, complete the following steps:
     - Disable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.
     - Inspect `~/.docker/config.json` to make sure that it does not contain `"credSstore": "osxkeychain"`, as some versions of Docker-for-mac may handle the setting correctly per https://github.com/docker/for-mac/issues/4192.
     - Authenticate with the Docker registry you intend to upload to.
     - Run the command:

       ```
       docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
       ```

     - Once the upload of images is successful you may enable **Docker > Preferences... > Securely store Docker logins in the macOS keychain**.

   Docker authentication notes:
   - Both HTTPS and HTTP are supported. Best practice for Docker registry security is to use HTTPS by utilizing standard Docker load tools, to ensure your images and platform are protected. However, when necessary you can use HTTP by specifying the `--tls-verify=false` flag.
   - When using a Docker registry on `localhost`, you might encounter that the image-tool Docker container is on the Docker network and typically without access to `localhost`. You can workaround this issue by using the `--network`

`host` argument to the Docker command. Note that `--network host` is a Docker argument, not an image-tool argument.

- Example of using `--network host` and `--tls-verify=false`:

```
$ docker run -d -p 5000:5000 --name registry registry:2
...
$ docker run --rm --network host apiconnect-image-tool-<version> upload localhost:5000 --tls-verify=false
...
```

Providing a certificate for verification, or disabling TLS verification

See the sample output of the tool usage for options to provide a certificate for verification or to disable TLS verification:

```
$ docker run --rm apiconnect-image-tool-<version> upload --help
upload docker images

Usage:
  image-tool upload REGISTRY [flags]

Flags:
      --cert-dir string   Directory with destination registry certificate tls.crt file
      --username string   User name
  -h, --help              help for upload
      --password          password for <username>
      --tls-verify        Verify TLS on destination registry (default true)

Global Flags:
      --accept-license   Accept the license for API Connect
      --debug            Enable debug logging
```

Notes:

- `--username=<username>` and `--password=<password>` can be used to specify credentials for authentication with the destination Docker registry.
- `--tls-verify=false` can be used to disable verification of the destination Docker registry certificate
- `--cert-dir <path>` can be used to provide a `tls.crt` file to be used for validation of the destination Docker registry certificate. For example:

```
docker run --rm -v <path-to-folder-with-tls.crt-file>:/cert apiconnect-image-tool-<version> upload <registry> --cert-dir /cert
```

6. Download the file IBM® API Connect *<version>* Operator Release Files for Containers
   a. Decompress the downloaded Operator Release Files for Containers

   Contents:

   - API Connect Operator operator custom resource definition (CRDs).
   - API Connect Operator Deployment and required resources CRDs.
   - API Connect Operator Deployment and required resources CRDs, for multiple-namespace installations.
   - DataPower Gateway Operator operator custom CRDs.
   - API Connect custom resource templates, and Certificate Manager

   The Operator Release Files for Containers files will be used later in deployment instructions.

   Note: The Operator Release files may include catalog and operator source for OpenShift. These files are not used when deploying on native Kubernetes.

   b. To access the API Connect custom resource templates and Certificate Manager, decompress the archive `helper_files.zip`.

   The zip file contains custom resource templates for:

   - Deployment of each subsystem.
   - Backup and restore of each subsystem
   - Custom certificates, both external and internal, for either standard deployments or 2-site HA deployments
   - Certificate support for multi-namespace deployments
   - Multi-site secret generation in a two data center deployment on Kubernetes.
   - Ingress Issuer and Subsystem Certificates Resources, for either standard deployments or 2-site HA deployments
   - Setting an administrator secret for DataPower Gateway

   The templates will be used later in deployment instructions.

What to do next

Continue with Deploying operators and cert-manager.

- **Verifying the integrity of IBM product files**
  You can verify the integrity of files to ensure that they originated from IBM and are not modified.

Verifying the integrity of IBM product files

You can verify the integrity of files to ensure that they originated from IBM and are not modified.

Before you begin

Import the public key so you can use it to verify files. You only need to import the public key once.

1. Install `gpg` (GNU Privacy Guard), available at https://www.gnupg.org/.
2. Create a file called PRD0003216key.pub.gpg and copy the following content into the file:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGP+Fr0BEADz5ppiT+RhdOw+PihTcMirn4umuinsTU66qpWZES+aLmSjw3Ik
BUlD0GQkJDn6qqZyMhYUwaLVicTtLrChykzfkd3Cdzis0WRsCJNdpuLIV9OS8L9a
zCMn09lHQgaf4+eYpDYOm5OBmRP+wqP3iJrC9dVho8oRaFUBmWi30Hak+YE8xBDU
bsigBCRuLxvtQJUTHXYK8lXdRQQz2acxTeo2+ozAsR594QZSCPuv9dBV93tKvbG0
mzLyrWUZKOACZuY5Uh2SVsvnwC5jZYd+KdaffzrdgR8/MFMKAIPsQNFExta5zaiN
ENhtwZd+XsUF7Jngij/oC30GI58WEP+iRHUduhx/olmqyVnOt+yeY+qtxCwss+88
azChYSdlGP540BqUZIcM1IvibbL8LqrCcOpOXCuHG/z44co1Ic1r9bveHYUUrhtv
kXS1fLQF3y44Oz+zf9rEah8CwmL5mQxqOeGszx+2bygxajqBJH28+H/MTMgbMsDJ
Y5M1PyojQLcv5+PO7vNs0w+IBWKzyJIFoSXI3jzGR53vXlyUJ3JgAGXQiIdo7LZl
Wa6WFoJkOnESRjfY00MRIoPdtvEJ99mzr09vwEvhEPsJfo0LkOGAkjjAq6E6Ihto
BBX+YlAASupnIuJBg9uV790YuCBPkDsQ+JrgIUTShE3Vu6MGMNfL3cxpxwARAQAB
tB1BUElDb25uZWN0IDxwc2lydEB1cy5pYm0uY29tPokCOgQTAQgAJAUCY/4WvQIb
DwULCQgHAgYVCgkICwIEFgIDAQIeAQUJAAAAAAAKCRACDta12+ZfO3RlD/9fl0cH
R68313LP9we/BSMjEzv+Ik3R8/nGlZ+huvekaNQp9nJxyhU3vkF89N6NTEzh7m78
m7dpRkwnJDNyd/AP7ACvqjqYSXUx49sw3ctyWxMqtiRaQb9ChR8vLIrybmQBoukQ
9p9h9DN+sNPz55AubZ/2kIu8mXDSgALyM3ULTqFRydAXtCNQ0/+iov443nvyyfSM
drfJOfi7xvxNnRGkubRnAC3Qyyyz/k/H9cb0HRmXKQxrxLwGuRiz0MrbYu9tqCJN
Zm9+Xs/AsW5R5iGu4aPpVIMayUWlU8fWhPmapqrN0JlGsfid38WXiCtYAqX8PbbT
vzy4ZDx6aJiPsKb66DuhmS5OJjvud480uTBaJBx7PKdqlL+VN7zjPQ8TKCO/qMcc
lLrBsVavQr1tLHi4EU8QzZyfIYvAMQZ9gL9FXqhCXUo1rz4qHytOikeK3XaStI+D
639LPfuiw+BJwDtLuJIaMVUyb+U/XBvCRGYmyZ8k3UUgDSj0wbtBHdXt4zSLdEhR
n1jegzLd248igap8bzDM0cWaji4YvvxLlOE2K9wWJj+cCU4+3PJuLv8wKmfIWp6j
Dv4il41RuROWkVJrj+U35B6dVGM7xOqk8WyFL548M7jimrs0gI+j2wxrKZtkqqOf
SzD8H00SJnD4HLhYmaatHuhiLJkXwB1wHeLO5g==
=1Zs0
-----END PGP PUBLIC KEY BLOCK-----
```

3. Run the following command to import the public key to `gpg`:

   ```
   gpg --import PRD0003216key.pub.gpg
   ```

   For example:

   ```
   $ gpg --import PRD0003216key.pub.gpg
   gpg: key 020ED6B5DBE65F3B: public key "APIConnect <psirt@us.ibm.com>" imported
   gpg: Total number processed: 1
   gpg:               imported: 1
   ```

4. Run the following command to verify that the key was imported successfully:

   ```
   gpg --list-key --fingerprint APIConnect
   ```

   Make sure that the output of the command matches the following information:

   ```
   $ gpg --list-key --fingerprint APIConnect
   pub   rsa4096 2023-02-28 [SCE]
         39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
   uid           [ unknown] APIConnect <psirt@us.ibm.com>
   ```

Procedure

Complete the following steps to verify product files for API Connect:

1. Visit IBM Fix Central and download the appropriate signatures_*<version>*.zip file for the version of API Connect that you will install or upgrade to. Extract the signatures_*<version>*.zip file to the folder containing the files that you want to verify.

2. Verify individual API Connect files to compare each file's signature with the fingerprint of the public key.
   To verify a file, run the following command:

   ```
   gpg --verify <file_to_verify>.asc
   ```

   For example, to verify helper_files.zip, ensure that it is stored in the same folder as the signatures file and run the following command:

   ```
   gpg --verify helper_files.zip.asc
   ```

   Example of a successful verification:

   ```
   $ gpg --verify helper_files.zip.asc
   gpg: assuming signed data in 'helper_files.zip'
   gpg: Signature made Mon 28 Sep 17:35:17 2020 PDT
   gpg:                using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
   gpg: Good signature from "APIConnect <psirt@us.ibm.com>" [unknown]
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: 39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
   ```

   Example of a failed verification:

   ```
   gpg: assuming signed data in 'helper_files.zip'
   gpg: Signature made Tue 11 Jul 2023 09:28:02 PM PDT
   gpg:                using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
   gpg: BAD signature from "APIConnect <psirt@us.ibm.com>" [unknown]
   ```

Deploying operators and cert-manager

Deploy the Kubernetes operator files and install cert-manager.

Before you begin

You must have completed the following installation tasks before you deploy the operators:

1. Review the installation requirements. See Deployment requirements.
2. Obtain the API Connect product files. See Obtaining product files.

About this task

Table 1 lists the operator that applies to each release of API Connect.

Table 1. Operators and operands for API Connect

API Connect release	API Connect Operator version	DataPower Operator version
10.0.7.0	5.1.0	1.9.0
10.0.6.0	5.0.0	1.7.0
10.0.5.3	3.3.0	1.6.6
10.0.5.2	3.2.0	1.6.3
10.0.5.1	3.1.0	1.6.2
10.0.5.0	3.0.0	1.6.0

Note: There can be only one instance of the API Connect operator in your Kubernetes environment. You cannot have multiple namespaces each with their own API Connect operator.
The deployment procedures differ depending on whether your deployment cluster uses a single namespace or multiple namespaces. Follow the instructions that apply to your deployment cluster:

- **Deploying operators in a single-namespace API Connect cluster**
  Deploy the Kubernetes operator files in a single namespace cluster.
- **Deploying operators in a multi-namespace API Connect cluster**
  Deploy the Kubernetes operator files in multi-namespace cluster.
- **Installing cert-manager and certificates in a two data center deployment**
  Additional instructions explaining how to install a cert-manager and certificates in a two data center disaster recovery deployment on Kubernetes.

Deploying operators in a single-namespace API Connect cluster

Deploy the Kubernetes operator files in a single namespace cluster.

Before you begin

- Ensure you completed the prerequisite tasks for installing operators. See Deploying operators and cert-manager.
- Be sure to review your strategy for using certificates with API Connect. See Deployment requirements.

Note: If you are deploying a multi-namespace API Connect cluster, do not use these instructions. Instead, go to Deploying operators in a multi-namespace API Connect cluster.

Procedure

1. Ensure **KUBECONFIG** is set for the target cluster:

   ```
   export KUBECONFIG=<path_to_cluster_config_YAML_file>
   ```

   An example path is **/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml**

2. Decide on a namespace to be used for installation.
   If not using the default namespace, create the namespace in the Kubernetes cluster where the API Connect deployment will take place. Use the following command, replacing **<namespace>** with the namespace for the deployment:

   ```
   kubectl create ns <namespace>
   ```

3. Install a cert-manager.
   Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, as long as the secret exists before deploying the CR, and the secret contains relevant certificate data. Typically, this is the tls.crt, tls.key, and ca.crt files. See Certificates in a Kubernetes environment.

   Restriction: Do not use this step to install a cert-manager if:
   - You want to use custom certificates. See Custom certificates on Kubernetes
   - You are deploying a two data center disaster recovery deployment on Kubernetes, see Installing cert-manager and certificates in a two data center deployment.

   After you complete one of the alternate sets of instructions in the links in this Note, continue with Step 4.

   a. Obtain the certificate manager.
      API Connect uses cert-manager v1.11.5 of cert-manager, which is a native Kubernetes certificate management controller.

      You can obtain cert-manager v1.11.5 from the API Connect v10 distribution **helper_files.zip** archive, or download it from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5

   b. Apply the CR:

      ```
      kubectl apply -f cert-manager-1.11.5.yaml
      ```

Do not specify a custom namespace.

See https://cert-manager.io/docs/release-notes/release-notes-1.11/.

c. Wait for `cert-manager` pods to enter `Running 1/1` status before proceeding. To check the status:

```
kubectl get po -n cert-manager
```

There are 3 `cert-manager` pods in total.

4. Create a registry secret with credentials to be used to pull down product images.
Use the following command, replacing **<namespace>** with the namespace for your deployment, and replacing **<registry_server>** with the location of the Docker Registry where the installation product images were pushed:

```
kubectl create secret docker-registry apic-registry-secret
--docker-server=<registry_server>
--docker-username=<username@example.com> --docker-password=<password> --docker-email=<username@example.com> -n <namespace>
```

5. Create a registry secret for the DataPower registry with the credentials to be used to pull down product images.
Use the following command, replacing **<namespace>** with the desired namespace for the deployment:

```
kubectl create secret docker-registry datapower-docker-local-cred
--docker-server=<docker_server> --docker-username=<username@example.com> --docker-password=<password> --docker-email=
<username@example.com> -n <namespace>
```

6. Create a DataPower admin secret.
Use the following command, replacing **<namespace>** with the desired namespace for the deployment:

```
kubectl create secret generic datapower-admin-credentials --from-literal=password=admin -n <namespace>
```

The admin secret will be used for **$ADMIN_USER_SECRET** when deploying the gateway CR.

7. If you are using a namespace other than `default`, open `ibm-apiconnect.yaml` in a text editor. Replace each occurrence of `default` with the namespace for your deployment.
8. Open `ibm-apiconnect.yaml` in a text editor. Replace the value of each `image:` key with the location of the apiconnect operator images (from the ibm-apiconnect container and the ibm-apiconnect-init container), either uploaded to your own registry or pulled from a public registry.

```
  serviceAccountName: ibm-apiconnect
   imagePullSecrets:
   - name: apic-registry-secret
   initContainers:
   - name: ibm-apiconnect-init
    image: <My_registry_or_public_registry>/ibm-apiconnect-operator-
init@sha256:0d3bbac7c67372ad013407a8049c69dbd08b1559b59a7606fcd87eb0f4519ce1
...
   containers:
    - name: ibm-apiconnect
    image: <My_registry_or_public registry>/ibm-apiconnect-
operator@sha256:a7473ee26c252fca4931b682cd95d73b4149e4b5b773834408f9f22d0246a76c
```

9. If DataPower Gateway is desired in this installation, and if you are using a namespace other than `default`, open `ibm-datapower.yaml` in a text editor. Replace each occurrence of `default` with the namespace for your deployment.
10. Open `ibm-datapower.yaml` in a text editor.
    a. Locate the `image:` key in the containers section of the deployment yaml immediately after `imagePullSecrets:`. Replace the value of the `image:` key with the location of the `datapower` operator image, either uploaded to your own registry or pulled from a public registry.
    b. Update the `IBM_DOCKER_HUB` and `IBM_ENTITLED_REGISTRY` value in the `env` section with the registry address where the images were uploaded with the image-tool in Obtaining product files.
    For example, the default entries are:

```
            - name: IBM_ENTITLED_REGISTRY
              value: "cp.icr.io/cp/datapower"
            - name: IBM_DOCKER_HUB
              value: "cp.icr.io/cp/datapower"
```

11. Install the ibm-apiconnect CRDs.

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

12. Install the ibm-apiconnect Kubernetes deployment.

```
kubectl apply -f ibm-apiconnect.yaml
```

13. If DataPower Gateway is desired in this installation, install the ibm-datapower Kubernetes deployment for DataPower Gateway, with the following command, replacing **<namespace>** with the desired namespace for the deployment (`-n <namespace>` may be omitted if default namespace is being used for installation):

```
kubectl apply -f ibm-datapower.yaml -n <namespace>
```

Troubleshooting: If your DataPower operator pod fails to start, see Troubleshooting install.

14. Install the `ingress-ca` Issuer to be used by `cert-manager`.
Note:
If you are installing a two data center disaster recovery deployment, skip this step, and follow the steps in: Installing cert-manager and certificates in a two data center deployment.

    a. Use the following command, replacing **<namespace>** with the desired namespace for the deployment:

```
kubectl apply -f ingress-issuer-v1.yaml -n <namespace>
```

The file **ingress-issuer-v1.yaml** is included in the release files contained in **helper_files.zip**

b. Validate that the command succeeded:

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

Example output indicating success:

```
NAME                          READY   SECRET                        AGE
analytics-ingestion-client    True    analytics-ingestion-client    70s
gateway-peering               True    gateway-peering               69s
gateway-service               True    gateway-service               69s
ingress-ca                    True    ingress-ca                    71s
portal-admin-client           True    portal-admin-client           71s
portal-tunnel-client          True    portal-tunnel-client          70s
```

The list of certificates might vary based on your deployment.

15. Continue with Installing the API Connect subsystems.

Deploying operators in a multi-namespace API Connect cluster

Deploy the Kubernetes operator files in multi-namespace cluster.

Before you begin

- Ensure you completed the prerequisite tasks for installing operators. See Deploying operators and cert-manager.
- Be sure to review your strategy for using certificates with API Connect. See Deployment requirements.

About this task

Note: If you are deploying a single-namespace API Connect cluster, do not use these instructions. Instead, go to Deploying operators in a single-namespace API Connect cluster.

- These instructions apply only to native k8s deployments. They do not apply to OpenShift deployments.
- The apiconnect operator is deployed as a cluster-scoped operator which watches every namespace in the cluster. The apiconnect operator can be installed in any namespace, although it is recommended that the API Connect operator is installed in its own dedicated namespace.
- The DataPower operator must be deployed in a namespace where gateway subsystem install is planned.
- Single ingress-ca certificate must be installed and used across the various namespaces where subsystems will be installed.

Procedure

1. Prepare your environment:
   a. Ensure **KUBECONFIG** is set for the target cluster:

   ```
   export KUBECONFIG=<path_to_cluster_config_YAML_file>
   ```

   An example path is **/Users/user/.kube/clusters/<cluster_name>/kube-config-<cluster_name>.yaml**

   b. Create namespaces on which subsystems are planned to be installed.
   The required namespaces are:

   - Apiconnect operator namespace
   - Gateway subsystem namespace
   - Management subsystem namespace
   - Portal subsystem namespace
   - Analytics subsystem namespace

   The following example values and deployment are used throughout these instructions:

   - An apiconnect operator is installed in **operator** namespace
   - A DataPower operator and gateway subsystem is installed in **gtw** namespace
   - The Management subsystem is installed in **mgmt** namespace
   - The Portal subsystem is installed in **portal** namespace
   - The Analytics subsystem is installed in **a7s** namespace

   c. Multi-namespace templates are provided in **helper_files.zip**, inside the **release_files.zip** you downloaded in Obtaining product files:

   ```
   helper_files_unpack/multi-ns-support/
   ```

2. Install cert-manager and configure certificates:
   Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, as long as the secret exists before deploying the CR, and the secret contains relevant certificate data. Typically, this is the tls.crt, tls.key, and ca.crt files. See Certificates in a Kubernetes environment.

   You can obtain cert-manager v1.11.5 from the API Connect v10 distribution **helper_files.zip** archive, or download it from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5.

   a. Install cert-manager v1.11.5, do not specify a custom namespace as cert manager will be installed in its own namespace.
   Restriction: Do not use Step 2.a to install a cert-manager if:
   - You want to use custom certificates. See Custom certificates on Kubernetes
   - You are deploying a two data center disaster recovery deployment on Kubernetes, see Installing cert-manager and certificates in a two data center deployment

i. `kubectl apply -f cert-manager-1.11.5.yaml`

ii. Wait for cert-manager pods to enter `Running 1/1` status before proceeding. Use the following command to check:

```
kubectl -n cert-manager get po
```

There are 3 cert-manager pods in total.

b. Install `ingress-ca` certificate:
   i. Locate `ingress-ca` certificate at `helper_files/multi-ns-support/ingress-ca-cert.yaml`
   ii. Create `ingress-ca` certificate in one of the subsystems. For example, if we choose `mgmt` namespace:

```
kubectl -n mgmt apply -f ingress-ca-cert.yaml
```

c. Install common issuers on all the namespaces:
   i. Locate `common-issuer.yaml` in `helper_files/multi-ns-support/`
   ii. Create `common-issuer.yaml` in all namespaces:

```
kubectl create -f common-issuer.yaml -n mgmt
kubectl create -f common-issuer.yaml -n gtw
kubectl create -f common-issuer.yaml -n ptl
kubectl create -f common-issuer.yaml -n a7s
```

d. Obtain the ingress-ca secret created by cert manager and create the secret on all namespaces:
Note: The instructions in this step result in having a `selfsigning-issuer` in every namespace. This issuer is typically referenced by an API Connect CR in the namespace with:

```
...
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
...
```

This issuer (and associated root certificate) does not need to be the same in every namespace, nor does it need to be the same for 2 CRs in the same namespace. Each subsystem can use its own `selfsigning-issuer`, because it is only used for certificates that are internal to the subsystem.

To ensure that the ingress-issuer is in sync across namespaces, it should be backed by the same `ingress-ca` certificate. This enables, for example, APIM to use a client certificate that matches the CA of the portal-admin ingress endpoint. The sync is achieved by copying around the `ingress-ca` certificate in each namespace, as shown in the following steps. It does not matter that it is the same issuer, or whether it has a different name. However it is important is that the ingress-ca is the same for the all the "ingress-issuer" that are going to be used by the subsystems.

   i. Make sure ingress-ca certificate READY status is set to true:

```
kubectl get certificate ingress-ca -n mgmt
NAME         READY   SECRET       AGE
ingress-ca   True    ingress-ca   9m24s
```

   ii. Cert-manager creates a secret called `ingress-ca` in `mgmt` namespace which represents the certificate we created in Step 2.c.
   iii. Use the following command to obtain the yaml format of the secret:

```
kubectl get secret ingress-ca -n <namespace-where-ingress-ca-cert-is-created> -o yaml > ingress-ca-secret-in-
cluster.yaml
```

   iv. Remove unwanted content from the secret yaml:

```
cat ingress-ca-secret-in-cluster.yaml | grep -v 'creationTimestamp' | grep -v 'namespace' | grep -v 'uid' | grep
-v 'resourceVersion' > ingress-ca-secret.yaml
```

   v. Create the secret using the YAML that you prepared in previous step on rest of the subsystem namespaces. In this example, in the `gtw`, `a7s` and `ptl` namespaces.

```
kubectl create -f ingress-ca-secret.yaml -n gtw
kubectl create -f ingress-ca-secret.yaml -n ptl
kubectl create -f ingress-ca-secret.yaml -n a7s
```

e. Install management certs in management namespace:
   i. Locate `management-certs.yaml` in `helper_files/multi-ns-support/`
   ii. Create `management-certs.yaml` in the `mgmt` namespace:

```
kubectl create -f management-certs.yaml -n mgmt
```

f. Install gateway certs in the gateway namespace:
   i. Locate `gateway-certs.yaml` in `helper_files/multi-ns-support/`
   ii. Create `gateway-certs.yaml` in the `gtw` namespace:

```
kubectl create -f gateway-certs.yaml -n gtw
```

3. Install the apiconnect operator:
   a. Install the ibm-apiconnect CRDs with the following command. Do not specify a namespace:

```
kubectl apply -f ibm-apiconnect-crds.yaml
```

   b. Create a registry secret with the credentials to be used to pull down product images with the following command, replacing `<namespace>` with the desired namespace for the apiconnect operator deployment.

```
kubectl -n <namespace> create secret docker-registry apic-registry-secret
              --docker-server=<registry_server>
              --docker-username=<username@example.com> --docker-password=<password>
              --docker-email=<username@example.com>
```

- For example, replace `<namespace>` with `operator`.

- **docker-password** can be your artifactory API key.
- **-n <namespace>** may be omitted if **default** namespace is being used for installation

c. Locate and open **ibm-apiconnect-distributed.yaml** in a text editor of choice. Then, find and replace each occurrence of **$OPERATOR_NAMESPACE** with **<namespace>**, replacing **<namespace>** with the desired namespace for the deployment. In this example, the namespace is **operator**.

d. Also in **ibm-apiconnect-distributed.yaml**, locate the **image:** keys in the containers sections of the deployment yaml right below **imagePullSecrets:**. Replace the placeholder values **REPLACE-DOCKER-REGISTRY** of the **image:** keys with the docker registry host location of the apiconnect operator image (either uploaded to own registry or pulled from public registry).

e. Install **ibm-apiconnect-distributed.yaml** with the following command

```
kubectl apply -f ibm-apiconnect-distributed.yaml
```

4. Install DataPower operator:
Deployment of the DataPower operator is only needed if you have at least one API Connect v10 Gateway subsystem (whether v5 compatible or not) to upgrade. If you are using DataPower in a different form factor such as an Appliance, you will not have an API Connect v10 Gateway subsystem to upgrade, and will not need the DataPower operator for your upgrade.

   Note: The DataPower Operator supports multiple instances of DataPower in the same cluster, separated by namespace. When deploying into multiple namespaces, ensure that any cluster-scoped resources are created with unique names for separate installations, such that they are not overwritten by subsequent installations. For example, DataPower cluster-scoped resources include **ClusterRoleBindings**.

   a. Create a registry secret for the DataPower registry with the credentials to be used to pull down product images with the following command, replacing **<namespace>** with the desired namespace for the deployment .(In this case **gtw**):

```
kubectl -n <namespace> create secret docker-registry datapower-docker-local-cred
              --docker-server=<registry_server>
              --docker-username=<username@example.com> --docker-password=<password>
              --docker-email=<username@example.com>
```

   - For example, replace **<*namespace*>** with **gtw**.
   - **docker-password** can be your artifactory API key.
   - **-n <namespace>** may be omitted if **default** namespace is being used for installation

   b. Create a DataPower admin secret, replacing **<namespace>** with the desired namespace for the deployment. For example, **gtw**. This secret will be used for **$ADMIN_USER_SECRET** later in the Gateway CR:

```
kubectl -n <namespace> create secret generic datapower-admin-credentials --from-literal=password=admin
```

   -n
   <namespace> may be omitted if **default** namespace is being used for installation.

   c. Locate and open **ibm-datapower.yaml** in a text editor of choice. Then, find and replace each occurrence of **default** with **<namespace>**, replacing **<namespace>** with the desired namespace for the deployment. For example, **gtw**.

   d. Install **ibm-datapower.yaml** with the following command:

```
kubectl -n <namespace> apply -f ibm-datapower.yaml
```

   Troubleshooting: If your DataPower operator pod fails to start, see Troubleshooting install.

5. Next, install the subsystems. Continue with Installing the API Connect subsystems.
Note: Unless otherwise stated, when performing the individual subsystem upgrades the **<*namespace*>** value of example **kubectl** commands for a given subsystem should be set to the namespace for the particular subsystem component they are acting on.

Installing cert-manager and certificates in a two data center deployment

Additional instructions explaining how to install a cert-manager and certificates in a two data center disaster recovery deployment on Kubernetes.

Before you begin

Before you install a cert-manager, you should know your strategy for using certificates with API Connect. Review the certificate requirements in Deployment requirements.

You should also have already completed Obtaining product files.

About this task

You must create certificates and keys for API Manager and the Developer Portal on both data centers, dc1 and dc2, and ensure that they match on both.

Use these instructions to install the supplied cert-manager and **ingress-issuer-v1.yaml**. Use these instructions if you are *not* using custom certificates.

Note that **ingress-issuer-v1.yaml** is supplied by cert-manager.

Note:
Do not use these instructions if you want to use custom certificates. See Custom certificates on Kubernetes.

Procedure

1. Install a cert-manager.
Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, on condition that the secret exists before you deploy the CR, and that the secret contains relevant certificate data. Typically, this is tls.crt, tls.key, and ca.crt files. See Certificates in a Kubernetes environment.

   a. Obtain the certificate manager.
   API Connect v10 uses cert-manager v1.11.5 of cert-manager, which is a native Kubernetes certificate management controller.

You can obtain cert-manager v1.11.5 from the API Connect v10 distribution **helper_files.zip** archive, or download it from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5

b. Apply the CR:

```
kubectl apply -f cert-manager-1.11.5.yaml
```

Do not specify a custom namespace.

See https://cert-manager.io/docs/release-notes/release-notes-1.5/.

c. Wait for **cert-manager** pods to enter **Running 1/1** status before proceeding. To check the status:

```
kubectl get po -n cert-manager
```

There are 3 **cert-manager** pods in total.

2. Use the following steps to allow **ingress-ca** secrets to be the same on both data centers.
   a. On DC1 apply the file **ingress-issuer-v1-dc1.yaml**:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc1.yaml
```

   b. Validate that the command succeeded:

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

   c. Export **ingress-ca** secret as a yaml from DC1:

```
kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
```

   d. Edit the **ingress-ca.yaml** file to remove all **annotations**, **labels**, **creationTimestamp**, **managedFields**, **manager**, **operation**, **time**, **resourceVersion**, **selfLink**, and **uid**. Also, if you are using a different **namespace** in DC2, then update the **namespace** field.
   e. Copy the **ingress-ca.yaml** from DC1 to DC2 and apply that file on DC2:

```
kubectl -n <namespace> apply -f ingress-ca.yaml
```

   f. On DC2 apply the file **ingress-issuer-v1-dc2.yaml**:

```
kubectl -n <namespace> apply -f ingress-issuer-v1-dc2.yaml
```

   g. Use the following commands to test that they are the same, on DC1 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d
> /tmp/ingress.pem.dc1
```

   h. On DC2 run:

```
kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}' | base64 -d
> /tmp/ingress.pem.dc2
```

   i. To see the differences run:

```
diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
```

The files should be the same.

   j. On DC2, to ensure that the certificates are working correctly and that they are using the **ingress-ca** secret. First, get the **portal-admin-client crt** file:

```
kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
/tmp/admin-client.crt
```

   k. Test that it is working by using OpenSSL:

```
openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
```

If it is working, you should see:

```
/tmp/admin-client.crt: OK
```

3. Continue with Deploying operators and cert-manager.

Installing the API Connect subsystems

Deploy the API Connect subsystem custom resources.

Before you begin

Attention: API Connect is not supported on a FIPS-enabled environment.
Before you install the subsystems, verify that you completed the following prerequisite tasks:

1. Obtaining product files
2. Deploying operators and cert-manager

About this task

Use the CR (custom resource) files to install the Management, Gateway, Analytics, and Portal subsystems for your API Connect cloud. CRs are extensions of the Kubernetes API. API Connect installs its subsystems using individual CRs. One CR is deployed per one managed subsystem. The API Connect operator manages the

lifecycle of the subsystem CRs and deploys and manages the microservice for each one.

CR templates are provided in the **helper_files** archive to install each API Connect subsystem. You uncompressed these files into an installation folder of your choosing in Obtaining product files.

- The API Connect operators and operands must be from the same version of API Connect. For example, the API Connect operator 10.0.4.0 does not successfully deploy the API Connect management subystem (operand) from Version 10.0.3.0.
- You can deploy multiple instances of API Connect into the same cluster by using different namespaces.
- Deploying multiple similar subsystems in a single namespace is not supported. For example, deploying two management subsystems, **mgmt1**, **mgmt2** in a single namespace where apiconnect operator is installed.
- For more information on Custom Resources in Kubernetes see https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/.
- For details on how the API Connect operator manages the CRs using the Kubernetes operator pattern, see https://kubernetes.io/docs/concepts/extend-kubernetes/operator/.
- For guidance on choosing object names and IDs in Kubernetes, see https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names.

Procedure

Complete each of the following instructions:

1. Installing the Management subsystem.
2. Installing the DataPower Gateway subsystem.
3. Installing the Developer Portal subsystem.
4. Installing the Analytics subsystem.
5. When all subsystems are running, verify you can access the API Connect Cloud Manager. Enter the Cloud Manager endpoint URL in your browser.
   Tip:
   You can find your Cloud Manager endpoint URL as follows:

   The property **cloudManagerEndpoint** is specified in the management CR (**management_cr.yaml**). When you edited the template, you specified a value for ***$STACK_HOST***:

   ```
   cloudManagerEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: admin.$STACK_HOST
         secret: cm-endpoint
   ```

   The template URL is **https://admin.<*STACK_HOST*>/admin**. The **/admin** suffix at the end is the name of the Cloud Manager administrator.

   For example, if ***$STACK_HOST*** is **myhost.subnet.example.com**, the Cloud Manager endpoint URL is:

   ```
   https://admin.myhost.subnet.example.com/admin
   ```

   The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See Accessing the Cloud Manager user interface.

What to do next

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

- **Installing the Management subsystem**
  Install the Management subsystem in your Kubernetes environment.
- **Installing the DataPower Gateway subsystem**
  Install the DataPower Gateway subsystem
- **Installing the Developer Portal subsystem**
  Install the Developer Portal subsystem.
- **Installing the Analytics subsystem**
  Install the IBM API Connect analytics subsystem.

Installing the Management subsystem

Install the Management subsystem in your Kubernetes environment.

Before you begin

Before you start this task, you should have already:

- Reviewed the Kubernetes requirements in Pre-installation requirements
- Completed Deploying operators and cert-manager

About this task

Edit the custom resource template for the Management system, apply the resource, verify that the pods are up and running, and verify that you can connect to the API Connect Cloud Manager.

Procedure

1. Edit the `management_cr` template CR, to replace the placeholders with values for your deployment.

   `$APP_PRODUCT_VERSION`
   API Connect application version for the subsystems.

   `version: <version_number>`

   Example version number: 10.0.7.0

   `$SECRET_NAME`
   Use for image pull.

   ```
   imagePullSecrets:
     - apic-registry-secret
   ```

   `$PROFILE`
   Specify your Management subsystem profile, where `n` indicates number of replicas, `c` number of cores, and `m` is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

   `$DOCKER_REGISTRY`
   The host name of the Docker Registry to which you uploaded the installation images. For example:

   `my.docker.registry.domain.example.com.`

   $INGRESS_CLASS
   The ingress class that you want the endpoint to use. This property is optional and if not specified, the ingress class with annotation `ingressclass.kubernetes.io/is-default-class: true` is used. If such an ingress class does not exist in the Kubernetes environment, then `nginx` is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.
   Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

   `$STACK_HOST`
   The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

   - Subdomain customization only
     Accept the prefixes predefined for the ingress hostnames to use and just replace all instances of `STACK_HOST` to be the desired ingress subdomain for the API Connect stack. For example, if your host is `myhost.subnet.example.com`:

     ```
     cloudManagerEndpoint:
          < ... >
       hosts:
       - name: admin.myhost.subnet.example.com
         secret: cm-endpoint

     apiManagerEndpoint:
          < ... >
       hosts:
       - name: manager.myhost.subnet.example.com
         secret: apim-endpoint

     platformAPIEndpoint:
          < ... >
       hosts:
       - name: api.myhost.subnet.example.com
         secret: api-endpoint

     consumerAPIEndpoint:
          < ... >
       hosts:
       - name: consumer.myhost.subnet.example.com
         secret: consumer-endpoint
     ```

   - Complete hostname customization
     Change both the predefined prefixes and the `STACK_HOST` subdomain to match your desired hostnames.

     For example, for `cloudManagerEndpoint`, you can replace `admin.$STACK_HOST` with `my.cloudmgr.myhost.subnet.example.com`, where `my.cloudmgr` replaces `admin`, and `myhost.subnet.example.com` replaces `STACK_HOST`. For example:

     ```
     cloudManagerEndpoint:
          < ... >
       hosts:
       - name: my.cloudmgr.myhost.subnet.example.com
         secret: cm-endpoint
     ```

     You can do this for some or all of the hostnames, depending on your customization requirements.

   `$STORAGE_CLASS`
   The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: `kubectl get sc`.

   `storageClassName: local-storage`

2. Edit the `license:` setting:
   a. Set `accept:` to `true` to accept the license. Note that the default value is `false`. If you do not accept the license, the Operator will not install the subsystem.
   b. Set `metric:` to track your product usage. Enter the unit of measure that is used for your program license:
      - PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used.
      - MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.

      For information on tracking monthly call volume, see Tracking API volume for auditing and compliance.

   c. Set `use:` to either `production` or `nonproduction`, to match the license you purchased.
   d. Set `license:` to the License ID for the version of API Connect that you purchased. See API Connect licenses.

Example entry to accept the license for a production system:

```
  license:
    accept: true
    metric: PROCESSOR_VALUE_UNIT
    use: production
    license: L-RJON-BZ5LSE
```

3. If installing with custom internal certificates, specify a site name.
   - The site name is used as the identifier for the PostgreSQL database cluster used by API Connect. This name should already have been decided upon and used to update the **custom-certs-internal.yaml** file as described in Generating custom certificates.
   - You must now add your chosen site name to **management_cr.yaml**, by creating a **siteName** property anywhere inside the **spec** block, set to the chosen site name:

     ```
       siteName: <site_name>
     ```

     Replace <*site_name*> with your chosen site name.

4. It is recommended to allocate 100Gi for write-ahead logging (WAL) storage.
   The default storage is:

   - One replica profile - 30Gi
   - Three replica profile - 47Gi

   For best performance, edit the CR to add the following entries, and set **volumeSize:** to 100Gi:

   ```
   spec:
     dbArchiveVolumeClaimTemplate:
       storageClassName: <storage-class>
       volumeSize: <volume-size>
   ```

5. Configure database backups for the management subsystem.
   Add the backup settings into the CR:

   ```
   spec:
     databaseBackup:
       host: <host>/<region>
       path: <path>
       credentials: <backup secret>
       backupCerts: <custom CA certificate>
       schedule: "<cron schedule>"
       repoRetentionFull: <number of days>
       backups3URIStyle: <path/host> # host or virtual style is the default
   ```

   Set the values as described in step 5 on Configuring management database backups.
   Note: If you do not have an S3 object-store available, you can install the management subsystem with no backup settings configured, by adding this annotation to the CR:

   ```
   metadata:
     annotations:
       apiconnect-operator/backups-not-configured: "true"
   ```

   If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
6. Optional: If you are installing as part of a two data center disaster recovery set up, complete the steps in Installing a two data center deployment on Kubernetes before you apply the edited file.
7. Install the management Custom Resource, replacing **<*namespace*>** with the target installation namespace in the Kubernetes cluster.

   ```
   kubectl apply -f management_cr.yaml -n <namespace>
   ```

8. Verify that the Management subsystem is fully installed:

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

   The installation has completed when the **READY** status is **True**, and the **SUMMARY** reports that all services are online (e.g. **9/9**). For example:

   ```
   NAME         READY   SUMMARY   VERSION    RECONCILED VERSION   AGE
   management   True    16/16      <version>   <version-build>      7m17s
   ```

9. Check your connection to the Cloud Manager user interface on the Management subsystem on your Cloud Manager endpoint.
   For example, if you accepted the default prefix of **admin**, and you set ***$STACK_HOST*** to **myhost.subnet.example.com**, the Cloud Manager endpoint URL is:

   ```
   https://admin.myhost.subnet.example.com/admin
   ```

   The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. For further details, see Accessing the Cloud Manager user interface.

10. Backup your management subsystem database and infrastructure configuration: Backing up and restoring the management subsystem.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Installing the DataPower Gateway subsystem

Before you begin

If you plan to install the Gateway subsystem in a remote cluster, be sure to satisfy the following requirements:

- Install both the API Connect and the DataPower operators.
  The versions of the operators on the remote cluster and the version of the Gateway operand must exactly match the versions used in the main cluster.
- Make sure the remote cluster has the certificates:
  - If you are using certificate manager, ensure that the ingress-ca is synced between sites.
  - If you are not using certificate manager, copy the client gateway certificates to the remote cluster and reference them in the Gateway CR.

About this task

Important: When you install the gateway subsystem, you must configure the gateway subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
Note: An alternate deployment scenario is to use a physical DataPower appliance for the gateway. If you are using an appliance for the gateway, do not install using API Connect Kubernetes CRs, since you do not need to configure a gateway subsystem in your Kubernetes cluster when using an appliance. For an appliance-based gateway, you must configure two endpoints in DataPower to be used as the Gateway Service Management Endpoint and the API invocation Endpoint. Enter these endpoints in the Configure Gateway Service screen in Cloud Manager. See Configuring DataPower Gateway for API Connect for instructions for configuring a DataPower appliance.
The installation folder where the **helper_files.zip** was extracted contains two templates for Gateway service. The **v5cgateway_cr.yaml** template is for the v5 compatible Gateway, and the **apigateway_cr.yaml** template is for the API Gateway.

Procedure

1. Edit the template CR, either **apigateway_cr.yaml** or **v5cgateway_cr.yaml**, to replace the placeholders with values for your deployment.

   $APP_PRODUCT_VERSION
   API Connect application version for the subsystems.

   ```
   version: <version_number>
   ```

   Example version number: 10.0.7.0

   $PROFILE
   Specify your gateway profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

   $SECRET_NAME
   Use for image pull.

   ```
   imagePullSecrets:
     - apic-registry-secret
   ```

   $DOCKER_REGISTRY
   The host name of the Docker Registry to which you uploaded the installation images. For example:

   ```
   my.docker.registry.domain.example.com.
   ```

   $INGRESS_CLASS
   The ingress class that you want the endpoint to use. This property is optional and if not specified, the ingress class with annotation **ingressclass.kubernetes.io/is-default-class:** **true** is used. If such an ingress class does not exist in the Kubernetes environment, then **nginx** is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.
   Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

   $STACK_HOST
   The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:
   - Subdomain customization only
     Accept the prefixes predefined for the ingress hostnames to use and just replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

     ```
     gatewayEndpoint:
           < ... >
         hosts:
         - name: rgw.myhost.subnet.example.com
           secret: gwv6-endpoint

       gatewayManagerEndpoint:
           < ... >
         hosts:
         - name: rgwd.myhost.subnet.example.com
           secret: gwv6-manager-endpoint
     ```

   - Complete hostname customization
     Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

     For example, for **gatewayEndpoint**, you can replace **rgw.$STACK_HOST** with **my.gateway.endpoint.myhost.subnet.example.com**, where **my.gateway.endpoint** replaces **rgw**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

     ```
     gatewayEndpoint:
           < ... >
         hosts:
     ```

```
    - name: my.gateway.endpoint.myhost.subnet.example.com
      secret: gwv6-endpoint
```

You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS

The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **kubectl get sc**.

```
storageClassName: local-storage
```

$ADMIN_USER_SECRET

Edit **$ADMIN_USER_SECRET** to set the value you created in Deploying operators and cert-manager.

```
adminUserSecret: $ADMIN_USER_SECRET
```

$PLATFORM_CA_SECRET

Set to **ingress-ca**. The **$PLATFORM_CA_SECRET** contains the CA certificate for the platform REST API endpoint. The gateway makes calls to the platform REST API and this property must be set so that the gateway can verify the server certificate.

```
mgmtPlatformEndpointCASecret:
  secretName: $PLATFORM_CA_SECRET
```

2. Edit the **license:** setting:
   a. Set **accept:** to **true** to accept the license. Note that the default value is **false**. If you do not accept the license, the Operator will not install the subsystem.
   b. Set **metric:** to track your product usage. Enter the unit of measure that is used for your program license:
      - PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used.
      - MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.

      For information on tracking monthly call volume, see Tracking API volume for auditing and compliance.

   c. Set **use:** to either **production** or **nonproduction**, to match the license you purchased.
   d. Set **license:** to the License ID for the version of API Connect that you purchased. See API Connect licenses.

   Example entry to accept the license for a production system:

```
  license:
    accept: true
    metric: PROCESSOR_VALUE_UNIT
    use: production
    license: L-RJON-BZ5LSE
```

3. Optional: If applicable to your deployment, you can override the resource settings for CPU and memory.
   - CPU override

     When you specify **spec.license.use: nonproduction** and a one replica profile **profile: n1xc4.m8**, the default CPU request for the gateway pod is 1 CPU. You can manually override this setting. For example, to request 2 CPUs:

```
template:
  - name: datapower
    containers:
    - name: gateway
      resources:
        requests:
          cpu: 2
```

     - When you specify **spec.license.use: production** and a one replica profile **profile: n1xc4.m8**, the default CPU request for the gateway pod is 4 CPUs.
     - The CPU limit cannot be specified separately from the CPU request. The value for the CPU request is used as both CPU request and CPU limit in the Gateway CR.

   - Memory override

     The default memory limit and request is 8Gi. The memory limit should always be greater than or equal to the memory request.

     For example, to request 10Gi and set a limit of 12Gi:

```
 template:
  - name: datapower
    containers:
    - name: gateway
      resources:
        limits:
          memory:  12Gi
        requests:
          cpu: 2
          memory: 10Gi
```

   Note: The override mechanism is specific only to the CPU and memory values, and cannot be used for other resource requests in the DataPower Gateway CR.

4. Enable or disable **syslogConfig**.

```
syslogConfig:
    enabled: false # if true, provide below details
    # remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
    # remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
    # secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

   - To disable syslog, remove the **syslogConfig** block from **apigateway_cr.yaml** and **v5cgateway_cr.yaml**. Alternatively, you can set **syslogConfig.enabled** to **false**.
   - To enable **syslogConfig** in the Gateways, set the following values in **apigateway_cr.yaml** and **v5cgateway_cr.yaml**:

     enabled

```
     enabled: true
```

     $DATAPOWER_SYSLOG_TCP_REMOTE_HOST

`remoteHost: <remote.host>`
String. The hostname of the server for `syslogConfig`.

$DATAPOWER_SYSLOG_TCP_REMOTE_PORT
`remotePort: <remote_port_number>`
Integer. The port number of the server for `syslogConfig`.

$DATAPOWER_SYSLOG_TCP_TLS_SECRET
`secretName: <secret>`
String. The TLS secret name of the server for `syslogConfig`.

5. Optional: If applicable to your deployment, update the GatewayCluster CR to configure `DataPowerService` APIs to customize your DataPower deployment. See Customizing a DataPower deployment.
6. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.
Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.
7. Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwksUrl`.

```
spec:
  ...
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the `jwksUrl`, describe the management CR and check the `status:` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.
8. Optional: If you want to use the In-cluster inter-subsystem communication feature, then uncomment the `mgmtPlatformEndpointSvcCASecret` section, and set the `secretName`:

```
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca  # Usually management-ca
```

Unless you customized your internal certificates, this is always `management-ca`.
9. Optional: Enable autoscaling of gateway pods to ensure high availability of DataPower pods. See Enabling autoscaling of gateway pods.
10. Install the Gateway Custom Resource by applying the Gateway template file:
For a DataPower API Gateway, run the following command:

```
kubectl apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
kubectl apply -f v5cgateway_cr.yaml -n <namespace>
```

11. To verify that the Gateway subsystem(s) are fully installed, run the following command:

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

The installation has completed when the `READY` status is `True`, and the `SUMMARY` reports that all services are online (`2/2`) for all the Gateway subsystems that were installed. Example:

```
NAME   READY   SUMMARY   VERSION     RECONCILED VERSION    AGE
gwv5   True    2/2       <version>   <version-build>       7m31s
gwv6   True    2/2       <version>   <version-build>       7m32s
```

There is no need to wait for the `READY` status to be `True` before moving on to the next Subsystem installation.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Installing the Developer Portal subsystem

Install the Developer Portal subsystem.

About this task

Important: When you install the portal subsystem, you must configure the portal subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
Important:

- To enable effective high availability for your Portal service, you need a latency that is less than 50 ms between all portal-db pods to avoid the risk of performance degradation. Servers with uniform specifications are required, as any write actions occur at the speed of the slowest portal-db pod, as the write actions are synchronous across the cluster of portal-db pods. It is recommended that there are three servers in each cluster of portal-db pods for the high availability configuration. The three servers can be situated in the same availability zone, or across three availability zones to ensure the best availability. However, you can configure high availability with two availability zones.
- The Portal does not do certain operations if the free space on certain volumes falls beneath predefined limits. In particular:
  The **databaseVolumeClaimTemplate** needs at least 4GB of free space to create new sites, restore site backups, or upgrade or change the URL of existing sites.

  The **webVolumeClaimTemplate** needs at least 256MB free space to create new sites, restore site backups, or upgrade or change the URL of existing sites.

- Ensure that your kernel or Kubernetes node has the value of its **inotify** watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a **non-ready** state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of **inotify** watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take **inotify** watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.
- Ensure that your kernel or Kubernetes node has a value of **nproc** (maximum number of processes) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow all of the Portal processes to execute. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls, you might need as many as 125205. If the number is too low, you will see errors like **"fork: retry: Resource temporarily unavailable"** in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.
- Ensure that your kernel or Kubernetes node has a value of **nofiles** (maximum number of open file descriptors) that applies to the user ID of the Portal pods that have been assigned, and that it is high enough to allow the Portal to open all of the files that it requires. For smaller installations this might be as low as 16384, but for larger installations that have more concurrent web calls and Portal web sites, you might need as many as 1048576. If the number is too low, you will see errors like **"too many open files"** in the Portal logs. Note that this value might need to be even higher if other, non-Portal, pods are sharing the same user ID.

The Portal endpoints values are used when you configure a Portal service in the Cloud Manager. See Registering a Portal service.

- **portalAdminEndpoint** - This is the Management Endpoint that is defined in Cloud Manager, which is used for communicating with API Manager.
- **portalUIEndpoint** - This is the Portal website URL defined in Cloud Manager. It determines the URL for the site that is created for each catalog. It is used for public access to the Portal from a browser.

Important: The endpoints defined here must be the ones used by the Management subsystem when registering the portal, and portal users when accessing portal sites. Do not attempt to proxy these endpoints. You will not be able to register the portal service nor access portal sites using any other endpoint URLs than those defined here.

Procedure

1. Edit the **portal_cr** template CR to replace the placeholders with values for your deployment.

   $APP_PRODUCT_VERSION
   API Connect application version for the subsystems.

   version: <*version_number*>

   Example version number: 10.0.7.0

   $PROFILE
   Specify your portal subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

   $SECRET_NAME
   Use for image pull.

   ```
   imagePullSecrets:
     - apic-registry-secret
   ```

   $DOCKER_REGISTRY
   The host name of the Docker Registry to which you uploaded the installation images. For example:

   my.docker.registry.domain.example.com.

   $INGRESS_CLASS
   The ingress class that you want the endpoint to use. This property is optional and if not specified, the ingress class with annotation **ingressclass.kubernetes.io/is-default-class: true** is used. If such an ingress class does not exist in the Kubernetes environment, then **nginx** is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.
   Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

   $STACK_HOST
   The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:
   - Subdomain customization only
     Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of **myhost.subnet.example.com**:

     ```
     portalAdminEndpoint:
         < ... >
         hosts:
         - name: api.portal.myhost.subnet.example.com
           secret: portal-admin

      portalUIEndpoint:
         < ... >
     ```

```
    hosts:
    - name: portal.myhost.subnet.example.com
      secret: portal-web
```

- Complete hostname customization
  Change both the predefined prefixes and the **$STACK_HOST** subdomain to match your desired hostnames.

  For example, for **portalAdminEndpoint**, you can replace **api.portal.$STACK_HOST** with **my.api.portal.myhost.subnet.example.com**, where **my.api.portal** replaces **api.portal**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

```
portalAdminEndpoint:
       < ... >
    hosts:
    - name: my.api.portal.myhost.subnet.example.com
      secret: cm-endpoint
```

  You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS
: The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **kubectl get sc**. For example, for local storage:

```
databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 12Gi
webVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
backupVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 300Gi
adminVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 20Gi
certVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 4Gi
```

$PLATFORM_CA_SECRET
: Set to **ingress-ca**. The **$PLATFORM_CA_SECRET** contains the CA certificate for the platform REST API endpoint. The portal makes calls to the platform REST API and this property must be set so that the portal can verify the server certificate.

```
mgmtPlatformEndpointCASecret:
  secretName: $PLATFORM_CA_SECRET
```

$CONSUMER_CA_SECRET
: Set to **ingress-ca**. The **$CONSUMER_CA_SECRET** contains the CA certificate for the consumer REST API endpoint. The portal makes calls to the consumer REST API and this property must be set so that the portal can verify the server certificate.

```
mgmtConsumerEndpointCASecret:
  secretName: $CONSUMER_CA_SECRET
```

Note:

- The **databaseVolumeClaimTemplate** has a minimum value of 30Gi. The **webVolumeClaimTemplate** has a minimum value of 20Gi. The **backupVolumeClaimTemplate** has a minimum value of 30Gi. However, all three need to be sized according to the installation. For more information on planning your portal PV requirements, see Portal subsystem.
- The **adminVolumeClaimTemplate** and **databaseLogsVolumeClaimTemplate** in this example have their minimum values and you do not need to change them.
- The **certVolumeClaimTemplate** is used by the nginx pod and in 2DCDR mode it is also by the **tunnel**, **db-remote** and **www-remote** pods. It can be sized small, just like the **adminVolumeClaimTemplate**, and does not need to be increased based on the number of sites.

2. Edit the **license:** setting:
   a. Set **accept:** to **true** to accept the license. Note that the default value is **false**. If you do not accept the license, the Operator will not install the subsystem.
   b. Set **metric:** to track your product usage. Enter the unit of measure that is used for your program license:
      - PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used.
      - MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.

      For information on tracking monthly call volume, see Tracking API volume for auditing and compliance.

   c. Set **use:** to either **production** or **nonproduction**, to match the license you purchased.
   d. Set **license:** to the License ID for the version of API Connect that you purchased. See API Connect licenses.

   Example entry to accept the license for a production system:

```
  license:
    accept: true
    metric: PROCESSOR_VALUE_UNIT
    use: production
    license: L-RJON-BZ5LSE
```

3. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties **mtlsValidateClient** and **jwksUrl**.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the **jwksUrl**, describe the management CR and check the **status:** section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.

4. Optional: If you want to use the In-cluster inter-subsystem communication feature, then uncomment the **mgmtPlatformEndpointSvcCASecret** and the **mgmtConsumerEndpointSvcCASecret** sections, and set the **secretName** for both:

```
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca  # Usually management-ca
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca  # Usually management-ca
```

Unless you customized your internal certificates, these secrets are always **management-ca**.

5. Optional: If you are installing as part of a two data center disaster recovery set up, complete the steps in Installing a two data center deployment on Kubernetes before you apply the edited file.
6. Install the Portal Custom Resource, replacing **<namespace>** with the target installation namespace in the Kubernetes cluster:

```
kubectl apply -f portal_cr.yaml -n <namespace>
```

7. Verify that the Portal subsystem is fully installed:

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

The installation is complete when **STATUS** reports **Running**, **Ready** reports that all services are online (**3/3**), and **MESSAGE** reports "Ready for API Cloud Manager service registration", for example:

```
NAME            READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE                                      AGE
portal          3/3     Running   <version>  <version>            Ready for API Cloud Manager service registration   2h
```

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

8. We highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.
The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource once the installation is complete. For example:

   a. Obtain the name of the encryptionSecret:

   ```
   kubectl get ptl portal -o yaml | grep encryptionSecret
   encryptionSecret: portal-enc-key
   ```

   The name of encryption secret is **portal-enc-key**.

   b. Save the secret yaml in a safe location:

   ```
   kubectl get secret portal-enc-key -o yaml > portal-enc-key.yaml
   ```

9. Backup the Portal installation. See Backing up and restoring the developer portal.
Note:
When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the **portal-www** pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.
To disable the endpoint check, complete the following update:

On Kubernetes, OpenShift, and IBM® Cloud Pak for Integration
Add the following section to the Portal custom resource (CR) template:

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

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you completed the installation of all the required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

- **Defining multiple portal endpoints for a Kubernetes environment**
Multiple public facing endpoints (portal-www) can be defined for the Developer Portal.

Defining multiple portal endpoints for a Kubernetes environment

Multiple public facing endpoints (portal-www) can be defined for the Developer Portal.

About this task

You can override the single endpoint definition for portal-WWW, and the associated portal-www-ingress TLS certificate, to support multiple portal-www endpoints.

For information about the endpoints for the Developer Portal, see Installing the Developer Portal subsystem.

Following are the example endpoints for configuring different sites served by the same Developer Portal service, as configured in this task:

- https://banking.example.com/loans
- https://insurance.example.com/vehicle

These unique endpoints allow Developer Portal sites to be defined on the Developer Portal service with different host names and domains. They replace endpoints that distinguish different sites by sub paths, as shown in the following examples:

- https://www.example.com/banking/loans
- https://www.example.com/insurance/vehicle

Procedure

Edit your `portal_cr` template. You can add multiple entries for your endpoints in the `hosts` element:
For example:

```
spec:
    portalUIEndpoint:
        annotations:
          cert-manager.io/issuer: ingress-issuer
        hosts:
        - name: ptl.host1.example.com
          secretName: portal-web-host1
        - name: ptl.host2.example.com
          secretName: portal-web-host2
        - name: ptl.host3.example.com
          secretName: portal-web-host3
```

Installing the Analytics subsystem

Install the IBM® API Connect analytics subsystem.

Review Planning your analytics deployment to ensure that you understand the choices for configuring the topology, disk space, and other elements of the analytics subsystem.

Note: When the analytics service is configured to store data, it uses OpenSearch, which requires map counts higher than the operating system defaults. The minimum recommended value is `262144`. If you plan to store analytics data locally (that is, do you not intend to disable local storage and offload all data), increase the default map count on every Kubernetes node:

1. To change the map counts on the live system, run the following command on every Kubernetes node:

   ```
   sudo sysctl -w vm.max_map_count=262144
   ```

2. To persist this change when node restarts occur, add the following setting to the /etc/sysctl.conf file:

   ```
   vm.max_map_count = 262144
   ```

For more information, see Important settings in the OpenSearch documentation.

- **Creating the analytics CR**
  Create the installation CR for API Connect analytics so that you can deploy the analytics subsystem.
- **Installing analytics**
  Deploy the analytics CR to install the analytics subsystem in your API Connect deployment.

Creating the analytics CR

Create the installation CR for API Connect analytics so that you can deploy the analytics subsystem.

Before you begin

Review Analytics preinstallation planning to decide on your deployment options.

About this task

To install the analytics subsystem, create a Custom Resource (CR) in a `.yaml` file. The file contains all of your configuration settings for the analytics deployment. Use the file to install, upgrade, and update the configuration of your analytics subsystem.
Note: The analytics_cr.yaml and analytics_dedicated_cr.yaml template files are stored in the directory where you extracted helper_files.zip.

Procedure

1. Create or copy the appropriate analytics CR file from the helper_files, depending on which deployment storage type you want, see: Analytics preinstallation planning.
   - If you want to use shared storage, use analytics_cr.yaml.
   - If you want to use dedicated storage, use analytics_dedicated_cr.yaml. Dedicated storage is only supported on three replica deployments.
2. Edit the CR file and update the following settings:
   You cannot install the analytics subsystem until the placeholders are replaced with real values for your environment.

   $APP_PRODUCT_VERSION
   API Connect application version for the subsystems.

   ```
   version: <version_number>
   ```

   Example version number: 10.0.7.0

   $PROFILE
   Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

   $SECRET_NAME
   The name of the secret used to pull images from the Docker registry.

   ```
   imagePullSecrets:
     - apic-registry-secret
   ```

   $DOCKER_REGISTRY
   The hostname of the Docker Registry to which you uploaded the installation images. For example:

   ```
   my.docker.registry.domain.example.com.
   ```

   $INGRESS_CLASS
   The ingress class that you want the endpoint to use. This property is optional and if not specified, the ingress class with annotation **ingressclass.kubernetes.io/is-default-class: true** is used. If such an ingress class does not exist in the Kubernetes environment, then **nginx** is used. If you do set this value, it must refer to a valid ingress class configured in your Kubernetes system.
   Note: This property is commented out in the template CR file. If you set this value, make sure to also uncomment it.

   $STACK_HOST
   The ingress subdomain for the API Connect stack. Used for the ingestion endpoint. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete hostname:
   - Subdomain customization only
     Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **$STACK_HOST** with the ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

     ```
     ingestion:
       endpoint:
          < ... >
         hosts:
         - name: ai.myhost.subnet.example.com
           secretName: analytics-ai-endpoint
           < .... >
     ```

   $STORAGE_CLASS
   The Kubernetes storage class to be used for persistent volume claims. For more information, see Analytics preinstallation planning. Find the available storage classes in the target cluster by running the following command: **kubectl get sc**.
   - For shared storage deployments:

     ```
     storage:
       type: shared
       shared:
         volumeClaimTemplate:
           storageClassName: local_storage
     ```

   - For dedicated storage deployments, set **$STORAGE_CLASS** in two places:

     ```
     storage:
       type: dedicated
       shared:
         volumeClaimTemplate:
           storageClassName: local_storage
           ...
       master:
         volumeClaimTemplate:
           storageClassName: local_storage
           ...
     ```

   $DATA_VOLUME_SIZE
   Replace **$DATA_VOLUME_SIZE** with the value that you calculated in Estimating storage requirements. If you are unable to estimate your storage requirement then set it to 500Gi.

   ```
   storage:
     ...
     shared:
       volumeClaimTemplate:
         ...
         volumeSize: 500Gi
   ```

3. Edit the **license** setting.
   a. Set **accept:** to **true** to accept the license. The default value is **false**. If you do not accept the license, the Operator does not install the subsystem.
   b. Set **metric:** to track your product usage. Enter the unit of measure that is used for your program license:
      - PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used.
      - MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.

For information on tracking monthly call volume, see Counting total API calls.

c. Set `use:` to either `production` or `nonproduction` to match the license you purchased.
d. Set `license:` to the license ID for the version of API Connect that you purchased. See API Connect licenses.

Example entry to accept the license for a production system:

```
  license:
    accept: true
    metric: PROCESSOR_VALUE_UNIT
    use: production
    license: L-RJON-CEBL97
```

4. Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the `jwksUrl`, describe the management CR and check the `status:` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.
Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.

5. Save the file.

What to do next

Install your analytics subsystem: Installing analytics.
Important: All analytics advanced configuration options can be added, changed, or removed after installation **except** for disablement or enablement of internal storage. If you want to disable internal storage, you must configure it in the analytics CR before installation: Disable local storage.

Installing analytics

Deploy the analytics CR to install the analytics subsystem in your API Connect deployment.

Before you begin

Now that you have a complete analytics CR file, double-check it for accuracy. In particular, verify that:

- No placeholder values remain in the analytics_cr.yaml file.
- You configured all the topology options that you plan to deploy.
- You configured appropriate volume size for storage.

Procedure

1. Run the following command to apply the analytics_cr.yaml:

```
kubectl apply -f path/to/analytics-cr -n namespace
```

where:
- *`path/to/analytics-cr`* is the path to your saved analytics_cr.yaml file.
- *`namespace`* is the name of your namespace.

2. Verify that the analytics subsystem is fully installed by running the following command:

```
kubectl get AnalyticsCluster -n namespace
```

where *`namespace`* is the name of your namespace.
The installation is complete when the `READY` status is `True` and the `SUMMARY` reports that all services are online. Example:

```
NAME        READY   SUMMARY   VERSION     RECONCILED VERSION    AGE
analytics   True    5/5       <version>   <version-build>       7m39s
```

The `SUMMARY` total count is based on your configured topology. Both numbers are equal when the installation is complete.

Results

Table 1 lists the pods that are expected in the three replica analytics deployment profile.

Table 1. Expected analytics subsystem pods

Expected	Pods	Note
3	director	Always expected.

Expected	Pods	Note
3	ingestion	Always expected.
3	mtls-gw	Always expected.
3	storage	Not expected if you disabled internal storage.
3	storage-os-master	Not expected if you disabled internal storage. Not expected if shared storage is configured.
3	osinit	Expected to be in completed state.

The one replica profile has 1 replica (instead of 3) of each pod.

What to do next

Take a backup of your analytics subsystem and configure scheduled analytics database backups: Analytics backup and restore.

Installing a two data center deployment on Kubernetes

Additional installation instructions for a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Before you begin

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see Two data center warm-standby deployment on Kubernetes and OpenShift.

Download the operator files and custom resource definitions for use during installation onto both of your data centers. Ensure that you complete the instructions for Deploying operators and cert-manager, and set the same ingress-ca certificate in both data centers: Installing cert-manager and certificates in a two data center deployment.

Familiarize yourself with the instructions in Installing the API Connect subsystems. Follow these instructions along with the 2DCDR specific steps in this topic.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see Backup and restore requirements for a 2DCDR deployment.
Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a 2DCDR configuration (Two data center warm-standby deployment on Kubernetes and OpenShift).

The following endpoints must be the same on both data centers:

Management subsystem endpoints

- `cloudManagerEndpoint`
- `apiManagerEndpoint`
- `platformAPIEndpoint`
- `consumerAPIEndpoint`

Portal subsystem endpoints

- `portalAdminEndpoint`
- `portalUIEndpoint`

For more information about how to set the endpoints, see Installing the Management subsystem on Kubernetes, Installing the Developer Portal subsystem on Kubernetes.

About this task

A 2DCDR deployment differs from a standard API Connect deployment in that the custom resources (CRs) for both the management and portal in each data center include a `multiSiteHA` configuration section. The `multiSiteHA` section is used to define which data center is the active and which is the warm-standby.

The following examples show `multiSiteHA` sections in the `PortalCluster` CRs for the active and warm-standby data centers.

```
siteName: dallas
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationdallas.cluster1.example.com
      secretName: dallas-ptl-replication-worker-1
  replicationPeerFQDN:
ptlreplicationraleigh.cluster2.example.com
  tlsClient:
    secretName: ptl-replication-client
...
```

```
siteName: raleigh
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationraleigh.cluster2.example.com
      secretName: raleigh-ptl-replication-worker-1
  replicationPeerFQDN:
ptlreplicationdallas.cluster1.example.com
  tlsClient:
    secretName: ptl-replication-client
...
```

Table 1. `multiSiteHA` configuration properties

Configuration property	Description
`mode`	The state of the data center in the context of the 2DCDR deployment. Valid values are: • `active` - indicates the active data center. • `passive` - indicates the warm-standby data center.

Configuration property	Description

`replicationEndpoint`
`  annotations:`
`  cert-manager.io/issuer:`
`ingress-issuer`
`  hosts:`
`  - name:`
`  secretName:`
	Contains details of the external ingress name for the subsystem in the current data center in the 2DCDR deployment.
	The hosts `name` is a unique fully qualified hostname that the other data center uses to communicate with the current data center.

`replicationPeerFQDN`	The ingress hostname for the other data center in the 2DCDR deployment. This information is required so that the two data centers can communicate with each other.
`tlsClient:` `  secretName`	The TLS client secret for the subsystem.

The `siteName` property is a unique descriptive name for the portal and management subsystems, in each data center. For example, `dallas` in data center 1 (DC1), and `raleigh` in data center 2 (DC2).

- Set a different `siteName` on the active to the one set on the warm-standby data center.
- `siteName` is used in the hostnames, and so can contain only `a-z` and `0-9` characters.
- You can set a `siteName` in the `ManagementCluster` or `PortalCluster` CRs only on installation. You cannot change the name after deployment. If you don't set a `siteName` at first deployment, an automated `siteName` is created for you.

To install a 2DCDR deployment:

1. Set the placeholder properties in the CR file for management and portal subsystems as detailed in Installing the API Connect subsystems, but do not run the final `kubectl`
   `apply` operation. Complete the additional 2DCDR configuration steps before you run `kubectl apply`.
2. Set the properties in the `multiSiteHA` section, and synchronize encryption secrets between the data centers as described in the procedure below:

Procedure

- **Installing to an active data center**
  The following example shows how to set the multi-site HA CR values for deploying to the active data center. In this example, the initial active data center is called `dallas` (DC1) and the warm-standby data center is called `raleigh` (DC2).

  1. Set Dallas to be active for the API Manager service on DC1.
     Edit the `ManagementCluster` CR file management_cr for DC1, and set the `multiSiteHA` properties as shown:

     ```
     siteName: dallas
     multiSiteHA:
       mode: active
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: ingress-issuer
         hosts:
         - name: mgrreplicationdallas.cluster1.example.com
           secretName: dc1-mgmt-replication
       replicationPeerFQDN: mgrreplicationraleigh.cluster2.example.com
       tlsClient:
         secretName: dc1-mgmt-replication-client
     ```

  2. Set Dallas to be active for the Developer Portal service on DC1.
     Edit the `PortalCluster` CR file portal_cr for DC1, and set the `multiSiteHA` properties as shown:

     ```
     multiSiteHA:
       mode: active
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: ingress-issuer
         hosts:
         - name: ptlreplicationdallas.cluster1.example.com
           secretName: dallas-ptl-replication-worker-1
       replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com
       tlsClient:
         secretName: ptl-replication-client
     ...
     siteName: dallas
     ```

  3. Create a secret called `ptl-encryption-key` on the active and warm-standby data centers.
     The secret must use the same random string in both data centers. Run the following command:

     ```
     kubectl create secret generic ptl-encryption-key --from-literal=encryption_secret=<RANDOM STRING> -n <namespace>
     ```

     The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

  4. When the secrets are created on both DC1 and DC2, update the portal CR on DC1 and DC2 and add the following under the `spec` section:

     ```
      encryptionSecret:
         secretName: ptl-encryption-key
     ```

     Warning: Ensure that you add the secret directly under the `spec` section and not inside the `multiSiteHA` section.
  5. Create a secret called `mgmt-encryption-key` on the active and warm-standby data centers.
     The secret must use the same random string in both data centers. Run the following command:
     Note: Do not use the same random string that you used to create the secret for `ptl-encryption-key`.

     ```
     kubectl create secret generic mgmt-encryption-key --from-literal=encryption_secret.bin=<RANDOM STRING> -n <namespace>
     ```

     The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

  6. When the secrets are created on both DC1 and DC2, update the portal CR on DC1 and DC2 and add the following under the `spec` section:

```
 encryptionSecret:
    secretName: mgmt-encryption-key
```

Warning: Ensure that you add the secret directly under the **spec** section and not inside the **multiSiteHA** section.

7. Apply the CR files to the subsystems in DC1.
   For example, to apply the **ManagementCluster** CR file to the DC1 cluster, run the following command:

   ```
   kubectl apply -f management_cr.yaml -n <namespace>
   ```

   Where ***<namespace>*** is the target installation namespace in the Kubernetes cluster for DC1. Repeat this process to apply the updated **PortalCluster** CR file to DC1.
   You can verify that the installation process is running with the following command:

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

   Note: Until the warm-standby is installed and running, the **haStatus** reports **PeerNotReachable**.
   Change ***ManagementCluster*** to ***PortalCluster*** to verify the Developer Portal installation. For more information, see Installing the API Connect subsystems.

8. Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

   Management subsystem endpoints
   - **cloudManagerEndpoint**
   - **apiManagerEndpoint**
   - **platformAPIEndpoint**
   - **consumerAPIEndpoint**

   Portal subsystem endpoints
   - **portalAdminEndpoint**
   - **portalUIEndpoint**

- **Installing the warm-standby data center**
  The following example shows how to set the multi-site HA CR values for the remote data center called **raleigh** (DC2) to be the warm-standby. Use the same encryption key on both sites.

  Ensure that the installation of the active data center is complete before you start the installation of the warm-standby data center.

  1. Set Raleigh to be warm-standby for the API Manager service on DC2.
     Edit the **ManagementCluster** CR file management_cr for DC2, and set the **multiSiteHA** properties as shown:

     ```
     multiSiteHA:
       mode: passive
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: ingress-issuer
         hosts:
         - name: mgrreplicationraleigh.cluster2.example.com
           secretName: raleigh-mgr-replication-worker-1
       replicationPeerFQDN: mgrreplicationdallas.cluster1.example.com
       tlsClient:
         secretName: mgr-replication-client
     ...
     siteName: raleigh
     ```

  2. Set Raleigh to be warm-standby for the Developer Portal service on DC2.
     Edit the **PortalCluster** CR file portal_cr for DC2, and set the **multiSiteHA** properties as shown:

     ```
     multiSiteHA:
       mode: passive
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: ingress-issuer
         hosts:
         - name: ptlreplicationraleigh.cluster2.example.com
           secretName: raleigh-ptl-replication-worker-1
       replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com
       tlsClient:
         secretName: ptl-replication-client
     ...
     siteName: raleigh
     ```

  3. Apply the CR files to the subsystems in DC2.
     For example, to apply the **ManagementCluster** CR file to the DC2 cluster, run the following command:

     ```
     kubectl apply -f management_cr.yaml -n <namespace>
     ```

     Where ***<namespace>*** is the target installation namespace in the Kubernetes cluster for DC2. Repeat this process to apply the updated **PortalCluster** CR file to DC2.
     You can verify that the installation process is running with the following command:

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

     When installation is complete, **haStatus** reports **Ready**.
     Change ***ManagementCluster*** to ***PortalCluster*** to verify the Developer Portal installation. For more information, see Installing the API Connect subsystems.

- **Validating your two data center deployment**
  While the management subsystems on the warm-standby and active data centers are synchronizing their databases, the management status reports **Warning**, and the **haStatus** reports **pending**:

```
kubectl get mgmt -n <namespace>

NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in
CR for details   8m59s

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

When the management database replication between sites is complete, the management status reports **Running**, and **status.haStatus** reports **Ready**:

```
oc get mgmt -n <namespace>

NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Running   10.0.7.0-0   10.0.7.0-0           Management is ready. HA status Ready - see HAStatus in CR
for details              8m59s

oc get mgmt -n <namespace> -o yaml

...
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the **haStatus** output shows the reason, for example::

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in
CR for details   8m59s

status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working",
      "reason": "na",
      "status": "True",
      "type": "PeerNotReachable"
    }
```

If the warning persists, see Troubleshooting a two data center deployment.
You can validate that your portal deployments are synchronizing by running **kubectl get pods** on both the active and warm-standby data centers, and confirming that the number and names of the pods all match (the UUIDs in the names might be different on each site), and that all are in the **Ready** state.

For additional replication verification checks, see Verifying replication between data centers. It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See Failing over to the warm-standby.

- **Converting a single data center to a two data center deployment on Kubernetes**
  Instructions on how to convert a single data center to a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Related concepts

- Maintaining a two data center deployment

Converting a single data center to a two data center deployment on Kubernetes

Instructions on how to convert a single data center to a two data center disaster recovery (2DCDR) deployment on Kubernetes.

Before you begin

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see Two data center warm-standby deployment on Kubernetes and OpenShift.

Download the operator files and custom resource definitions for use during installation onto both of your data centers. Ensure that you complete the instructions for Deploying operators and cert-manager, and set the same ingress-ca certificate in both data centers: Installing cert-manager and certificates in a two data center deployment.

Familiarize yourself with the instructions in Installing the API Connect subsystems. Follow these instructions along with the 2DCDR specific steps in this topic.

Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see Backup and restore requirements for a 2DCDR deployment.
Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a 2DCDR configuration (Two data center warm-standby deployment on Kubernetes and OpenShift).

About this task

You can convert a single data center API Connect to a 2DCDR deployment. Both data centers must have the same **portal-admin** and **portal-www** endpoints. The main difference is that the custom resource (CR) files for both the Management subsystem and the Developer Portal subsystem on each data center must include a **multiSiteHA** configuration section, and it's this section that determines the two data center DR deployment. It's also this section that's used to control the deployment in the future, for example in failover and recovery situations.

The following codeblocks show examples of completed **multiSiteHA** sections in the **PortalCluster** CR files for two data centers, one active and one warm-standby. Both the **ManagementCluster** and the **PortalCluster** CR templates have a **multiSiteHA** section containing the placeholder **$MULTI_SITE_HA**, and it is this placeholder that must be replaced with the values for your deployment.

```
siteName: dallas
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationdallas.cluster1.example.com
      secretName: dallas-ptl-replication-worker-1
  replicationPeerFQDN:
ptlreplicationraleigh.cluster2.example.com
  tlsClient:
    secretName: ptl-replication-client
...
```

```
siteName: raleigh
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationraleigh.cluster2.example.com
      secretName: raleigh-ptl-replication-worker-1
  replicationPeerFQDN:
ptlreplicationdallas.cluster1.example.com
  tlsClient:
    secretName: ptl-replication-client
...
```

The configuration properties are described in the following table.

Table 1. The configuration properties of the multi-site HA section in the CR file

Configuration property	Description
mode	The state of the data center in the context of the two data center deployment. Valid values are: • **active** - indicates the primary data center • **passive** - indicates the warm-standby data center
replicationEndpoint **annotations:** **cert-manager.io/issuer** **hosts:** **- name:** **secretName:**	Contains details of the external ingress name for the subsystem in the current data center in the two data center deployment. The hosts **name** is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
replicationPeerFQDN	The ingress hostname for the other data center in the two data center deployment. This information is required so that the two data centers can communicate with each other.
tlsClient: **secretName**	The TLS client secret for the subsystem.
siteName	A descriptive name for the Portal subsystem, and the Management subsystem, in the current data center, for example **dallas**. This name is used in the hostnames, and so can contain only **a-z** and **0-9** characters. Note that you can set a **siteName** in the **ManagementCluster** or **PortalCluster** CR only when the subsystems are first deployed, and you cannot change the name after deployment. If you don't set a **siteName** at first deployment, an automated **siteName** is created for you. For example, if you are converting an existing single Management cluster to a two data center Management cluster by adding the multi-site HA values, you will not be able to configure a **siteName** for the existing Management cluster.

To install a two data center disaster recovery deployment, you must set all of the placeholder properties in the CR file for both the Management and the Developer Portal subsystems as detailed in the relevant topics in the Installing the API Connect subsystems section, as well as the properties in the multi-site HA section that are detailed here. Some examples of complete CR files are shown in the Example section.

Procedure

- **Converting a single data center deployment to two data centers**
  When converting a single data center deployment into a two data center deployment, it is best practice to use the existing data center as the active one, and add a new warm-standby data center elsewhere. This practice ensures that the data is retained.
  Remember:
  - Both data centers must have the same **portal-admin** and **portal-www** endpoints.
  - You cannot configure a **siteName** for the existing Management or Portal cluster because the **siteName** property can be configured only at first deployment.

  The following example shows how to update the multi-site HA CR values for making the local data center called **dallas** (DC1) active, and for setting the new remote data center called **raleigh** (DC2) as warm-standby.

- Install a cert-manager.
  Use of a certificate manager adds convenience to the generation and management of certificate, but is not required. Whenever a custom resource (CR) takes a certificate secret name as input, you can point to any secret name, on condition that the secret exists before you deploy the CR, and that the secret contains relevant certificate data. Typically, this is tls.crt, tls.key, and ca.crt files. See Certificates in a Kubernetes environment.
  Note: Run these steps on DC1 only if you don't already have cert-manager running.
  1. Obtain the certificate manager.

API Connect v10 uses cert-manager v1.11.5 of cert-manager, which is a native Kubernetes certificate management controller.

You can obtain cert-manager v1.11.5 from the API Connect v10 distribution **helper_files.zip** archive, or from https://github.com/cert-manager/cert-manager/releases/tag/v1.11.5.

Note: Cert-manager is bundled as a common service in IBM CloudPack for Integration.

2. Apply the CR:

   ```
   kubectl apply -f cert-manager-1.11.5.yaml
   ```

   Do not specify a custom namespace.

   See https://docs.cert-manager.io/en/release-0.10/getting-started/install/kubernetes.html.

3. Wait for **cert-manager** pods to enter **Running 1/1** status before proceeding. To check the status:

   ```
   kubectl get po -n cert-manager
   ```

   There are 3 **cert-manager** pods in total.

- Use the following steps to allow **ingress-ca** secrets to be the same on both data centers.
  1. On DC1 apply the file **ingress-issuer-v1-dc1.yaml**:

     ```
     kubectl -n <namespace> apply -f ingress-issuer-v1-dc1.yaml
     ```

  2. Validate that the command succeeded:

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

  3. Export **ingress-ca** secret as a yaml from DC1:

     ```
     kubectl -n <namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
     ```

  4. Edit the **ingress-ca.yaml** file to remove all labels, annotations, **creationTimestamp**, **resourceVersion**, **uid**, and **selfLink**.
  5. Copy the **ingress-ca.yaml** from DC1 to DC2 and apply that file on DC2:

     ```
     kubectl -n <namespace> apply -f ingress-ca.yaml
     ```

  6. On DC2 apply the file **ingress-issuer-v1-dc2.yaml**:

     ```
     kubectl -n <namespace> apply -f ingress-issuer-v1-dc2.yaml
     ```

  7. Use the following commands to test that they are the same, on DC1 run:

     ```
     kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
     /tmp/ingress.pem.dc1
     ```

  8. On DC2 run:

     ```
     kubectl -n <namespace> get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
     /tmp/ingress.pem.dc2
     ```

  9. To see the differences run:

     ```
     diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
     ```

     The files should be the same.
  10. To ensure that the certificates are working correctly and that they are using the **ingress-ca** secret. First, get the **portal-admin-client crt** file:

      ```
      kubectl -n <namespace> get secrets portal-admin-client -o yaml | grep tls.crt | awk '{print $2}' | base64 -d >
      /tmp/admin-client.crt
      ```

  11. Test that it is working by using OpenSSL:

      ```
      openssl verify -verbose -CAfile /tmp/ingress.pem.dc1 /tmp/admin-client.crt
      ```

      If it is working, you should see:

      ```
      /tmp/admin-client.crt: OK
      ```

- **Installing to an active data center**
  The following example shows how to set the multi-site HA CR values for deploying to an active data center (DC). In this instance, we are deploying to a local data center called **dallas** (DC1) that has a remote data center called **raleigh** (DC2).

  1. Set Dallas to be active for the API Manager service on DC1.
     As the API Manager service already exists, you cannot set the **siteName** property. Add the multi-site HA section properties to the existing **ManagementCluster** CR file for DC1. For example:

     ```
     multiSiteHA:
       mode: active
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: ingress-issuer
         hosts:
         - name: mgrreplicationdallas.cluster1.example.com
           secretName: dc1-mgmt-replication
       replicationPeerFQDN: mgrreplicationraleigh.cluster2.example.com
       tlsClient:
         secretName: dc1-mgmt-replication-client
     ```

  2. Set Dallas to be active for the Developer Portal service on DC1.
     As the Developer Portal service already exists, you cannot set the **siteName** property. Add the multi-site HA section properties to the existing **PortalCluster** CR file for DC1. For example:

```
multiSiteHA:
  mode: active
  replicationEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationdallas.cluster1.example.com
      secretName: dallas-ptl-replication-worker-1
  replicationPeerFQDN: ptlreplicationraleigh.cluster2.example.com
  tlsClient:
    secretName: ptl-replication-client
```

3. Create a secret called `ptl-encryption-key` on the active and warm-standby data centers.
   You must create a secret on the active and warm-standby sites that use the same random string, run the following command:

   ```
   kubectl create secret generic ptl-encryption-key --from-literal=encryption_secret=<RANDOM STRING> -n <namespace>
   ```

   The string can consist of uppercase letters, lowercase letters, and numbers, and must be at least 64 characters and no more than 100 characters.

4. When the secrets are created on both DC1 and DC2, update the Portal CR on DC1 and DC2 to include in the `spec` object:

   ```
   encryptionSecret:
     secretName: ptl-encryption-key
   ```

   Warning: Ensure that you add the secret to the `spec` object and not to the `multiSiteHA` object.
5. Apply the CR files to the subsystems in DC1.
   For example, to apply the `ManagementCluster` CR file to the DC1 cluster, run the following command:

   ```
   kubectl apply -f management_cr.yaml -n <namespace>
   ```

   Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC1. Repeat this process to apply the updated `PortalCluster` CR file to DC1.
   You can verify that the updated CR files have been applied by running the following command:

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

   You can verify that the installation process is running with the following command:

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

   Note: Until the warm-standby is installed and running, the `haStatus` reports `PeerNotReachable`.
   Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see Installing the API Connect subsystems.

6. Set your dynamic router to direct all traffic to DC1.
   This includes setting the four endpoints for the API Manager cluster, and the two endpoints for the Developer Portal cluster.

- **Installing to a warm-standby data center**
  The following example shows how to set the multi-site HA CR values for the remote data center called `raleigh` (DC2), which is being installed as the warm-standby, standby location. You must use the same encryption key on both sites.

1. Get the encryption key from Dallas the `active` (DC1) site:
   a. Run the command `kubectl get mgmt -o yaml|grep enc` to get the db encryption key on the `active` site, for example:

      ```
      $ kubectl -n <namespace> get mgmt -o yaml|grep enc
      ```

      An example of possible output:

      ```
      encryptionSecret: dallas-enc-key
      ```

      Make a note of the secret name. In this case, it's `dallas-enc-key`.
   b. Run `kubectl get secret dallas-enc-key -o yaml`, an example of possible output:

      ```
      apiVersion: v1
      data:
        encryption_secret.bin:
      VKBNFj7sAOizxvE1H6i+9P31oJHvWWsO+x**********************************EGe/K+x6b3D7FEWGgoyGlWBUJKB4+T21My2iR5rBTov
      pyLiY5g*******************************************tSiRcQKegMPNBPgL829SVBCxuv3I=
      kind: Secret
      metadata:
        creationTimestamp: "2020-08-28T12:53:41Z"
        labels:
          app.kubernetes.io/instance: m1
          app.kubernetes.io/managed-by: ibm-apiconnect
          app.kubernetes.io/name: dallas-enc-key
        name: dallas-enc-key
        namespace: default
        resourceVersion: "43039"
        selfLink: /api/v1/namespaces/default/secrets/m1-enc-key
        uid: 46c92395-9cc2-4421-b2c4-48e472c0cbb1
      type: Opaque
      ```

   c. Copy the output that you got and put it into a new file. Delete the contents of `metadata:` and add back in a `name:` attribute, in this case `dc1-enc-key`. An example of the format is here:

      ```
      apiVersion: v1
      data:
        encryption_secret.bin:
      VKBNFj7sAOizxvE1H6i+9P31oJHvWWsO+x**********************************EGe/K+x6b3D7FEWGgoyGlWBUJKB4+T21My2iR5rBTov
      pyLiY5g*******************************************tSiRcQKegMPNBPgL829SVBCxuv3I=
      kind: Secret
      metadata:
        name: dc1-enc-key
      type: Opaque
      ```

d. Save the file. In this example, it's called `dc1-enc-key.yaml`.

2. On Raleigh the (DC2) site:
   a. Copy the yaml file that you just saved that contains the `encryption_secret.bin` from Dallas the `active` (DC1) site.
   b. Run `kubectl create -f dc1-enc-key.yaml` to create the same secret with the same key on the Raleigh (DC2) site. For example:

      ```
      $ kubectl  -n <namespace> create -f /tmp/dc1-enc-key.yaml
      ```

      An example of possible output:

      ```
      secret/dc1-enc-key created
      ```

   c. In the management CR for Raleigh the (DC2) site, add the following entry to the `spec`.

      ```
      encryptionSecret:
          secretName: dc1-enc-key
      ```

3. Set Raleigh to be warm-standby for the API Manager service on DC2.
   Edit the `ManagementCluster` CR file management_cr for DC2, and set the multi-site HA section properties, for example:

```
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manage.io/issuer: ingress-issuer
    hosts:
    - name: mgrreplicationraleigh.cluster2.example.com
      secretName: raleigh-mgr-replication-worker-1
  replicationPeerFQDN: mgrreplicationdallas.cluster1.example.com
  tlsClient:
    secretName: mgr-replication-client
...
siteName: raleigh
```

4. Set Raleigh to be warm-standby for the Developer Portal service on DC2.
   Edit the `PortalCluster` CR file portal_cr for DC2, and set the multi-site HA section properties, for example:

```
multiSiteHA:
  mode: passive
  replicationEndpoint:
    annotations:
      cert-manage.io/issuer: ingress-issuer
    hosts:
    - name: ptlreplicationraleigh.cluster2.example.com
      secretName: raleigh-ptl-replication-worker-1
  replicationPeerFQDN: ptlreplicationdallas.cluster1.example.com
  tlsClient:
    secretName: ptl-replication-client
...
siteName: raleigh
```

5. Ensure that you completed the steps that are in the **Installing to an active data center** section to create a secret called `ptl-encryption-key` on DC1 and DC2, and to update the Portal CR on DC1 and DC2 to include the `ptl-encryption-key` in the `spec` object.
6. Apply the CR files to the subsystems in DC2.
   For example, to apply the `ManagementCluster` CR file to the DC2 cluster, run the following command:

   ```
   kubectl apply -f management_cr.yaml -n <namespace>
   ```

   Where `<namespace>` is the target installation namespace in the Kubernetes cluster for DC2. Repeat this process to apply the updated `PortalCluster` CR file to DC2.
   You can verify that the updated CR files are applied by running the following command:

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

   You can verify that the installation process is running with the following command:

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

   When installation is complete, `haStatus` reports `Ready`.
   Change `ManagementCluster` to `PortalCluster` to verify the Developer Portal installation. For more information, see Installing the API Connect subsystems.

Related concepts

- Maintaining a two data center deployment

Troubleshooting installation on Kubernetes

Review the following known issues and troubleshooting tips if you encounter a problem while installing API Connect on Kubernetes

- DataPower operator fails to start

DataPower operator fails to start

There is a known issue on Kubernetes that can cause the DataPower operator to fail to start. In this case, the DataPower operator pods can fail to schedule, and display the status message: `no nodes match pod topology spread constraints (missing required label)`. For example:

```
0/15 nodes are available: 12 node(s) didn't match pod topology spread constraints (missing required label),
3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate.
```

You can workaround the issue by editing the DataPower operator deployment and reapplying it, as follows:

1. Delete the DataPower operator deployment, if deployed already:

   ```
   kubectl delete -f ibm-datapower.yaml -n <namespace>
   ```

2. Open `ibm-datapower.yaml`, and locate the `topologySpreadConstraints:` section. For example:

   ```
   topologySpreadConstraints:
   - maxSkew: 1
     topologyKey: zone
     whenUnsatisfiable: DoNotSchedule
   ```

3. Replace the values for `topologyKey:` and `whenUnsatisfiable:` with the corrected values that are shown in the following example:

   ```
   topologySpreadConstraints:
   - maxSkew: 1
     topologyKey: topology.kubernetes.io/zone
     whenUnsatisfiable: ScheduleAnyway
   ```

4. Save `ibm-datapower.yaml` and deploy the file to the cluster:

   ```
   kubectl apply -f ibm-datapower.yaml -n <namespace>
   ```

Deploying on OpenShift and Cloud Pak for Integration

You can install API Connect in an OpenShift environment or as the API Management capability in IBM Cloud Pak for Integration.

If you are deploying multiple API Connect instances in the same OpenShift cluster, all of them should be at the same version. The recommended configuration is to have a single operator in the global namespace, managing multiple API Connect instances in separate namespaces.

Attention: If you want to configure FIPS support for API Connect, you must enable FIPS on the cluster as part of the initial OpenShift deployment. You cannot modify an existing cluster to enable FIPS support. For more information, see Configuring FIPS support on OpenShift.

- **Configuring FIPS support on OpenShift**
  Configure support for FIPS (Federal Information Processing Standards) on the cluster where you will install IBM API Connect.
- **Preparing for installation**
  Set up your OpenShift environment in preparation for deploying API Connect.
- **Installation procedures with IBM Cloud Pak for Integration**
  The API Management capability of IBM Cloud Pak for Integration uses IBM API Connect.
- **Installing API Connect in a single namespace on OpenShift**
  Use individual subsystem CRs to install API Connect subsystems in a shared namespace on OpenShift.
- **Installing API Connect subsystems in different namespaces or environments on OpenShift**
  Install API Connect subsystems in different namespaces, for example Management subsystem in a namespace that is called `apicmgmt`, Gateway in a namespace called `apicgway`. These same steps can also be used for installing subsystems in different environments.
- **Installing with the top-level CR on OpenShift**
  Use the single, top-level `APIConnectCluster` CR to deploy API Connect on OpenShift.
- **Air-gapped installation**
  In a disconnected environment using Red Hat OpenShift Container Platform (OCP), install API Connect by mirroring product images to a bastion host or a portable device and then completing the installation.
- **Installing a two data center deployment with Cloud Pak for Integration**
  How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift with Cloud Pak for Integration.
- **Installing a two data center deployment on OpenShift**
  How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift.
- **Enable additional features post-install**
  After installing API Connect, you can enable additional features that are related to inter-subsystem communication.
- **Troubleshooting installation on OpenShift**
  Review the following known issues and troubleshooting tips if you encounter a problem while installing API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Configuring FIPS support on OpenShift

Configure support for FIPS (Federal Information Processing Standards) on the cluster where you will install IBM API Connect.

Before you begin

API Connect supports FIPS with the following limitations:

- API Connect must be installed on Red Hat OpenShift Container Platform 4.12 or 4.14
- Only FIPS version 140-2 is supported
- FIPS must be configured on the cluster before you install API Connect
- You must deploy API Connect 10.0.7.0 or later, with DataPower Gateway 10.5.3 or later
  For DataPower, refer to the following documentation for the most up-to-date information on DataPower FIPS support:
  - Known limitations and restrictions
  - Cloud Pak for Integration - Gateway instance additionally notes the FIPS and FIPS-Wall status

About this task

To be FIPS compliant, an organization must adhere to the various data security and computer system standards outlined in the FIPS requirements. FIPS compliance for API Connect is supported by implementing a "FIPS wall".

What is FIPS?

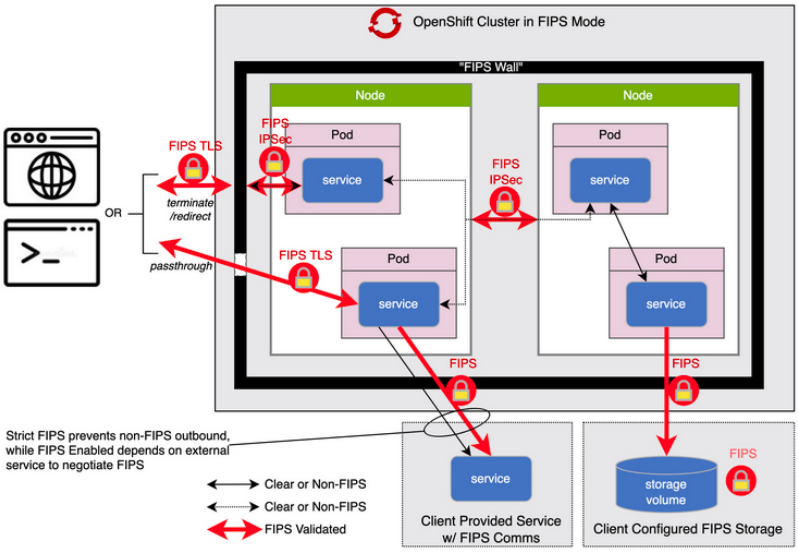
Federal Information Processing Standards (FIPS) are standards and guidelines issued by the National Institute of Standards and Technology (NIST) for federal government computer systems. The standards are developed when there are compelling federal government requirements for standards, such as for security and interoperability, but acceptable industry standards or solutions do not exist. Government agencies and financial institutions use these standards to ensure that products conform to specified security requirements.

API Connect supports version 140-2 of the FIPS requirements, implemented using the "FIPS wall" approach.

What is a FIPS wall?

The "FIPS wall" is a boundary approach to FIPS compliance that is used by IBM products. All pods in the cluster are FIPS-tolerant (the pods can run without issues on a FIPS-enabled OpenShift cluster), while creating a compliant "boundary" that is secured at external points of contact (known as "touch points").

Traffic inside the boundary is secure because the communication between nodes is automatically encrypted at the OpenShift Container Platform level using the IPSec protocol. Traffic within any node happens in-memory, so it never leaves the node. The following diagram illustrates a typical OpenShift cluster that is configured to support the FIPS wall. In the diagram, the "wall" is represented with a heavy black border. Communications between pods within the border are secured with IPSec encryption, and communications that cross the border to external devices are secured using other means for example, TLS encryption).



What are the cluster requirements to support a FIPS wall?

- FIPS is enabled on the cluster during installation (by setting `fips: true` in the install-config.yaml file)
- FIPS is enabled for node-to-node communication (using the OVN-Kubernetes Container Network Interface cluster network provider, and with IPSec enabled)
- The etcd key-value store is encrypted with `aescbc`
- Storage is encrypted with FIPS ciphers
- Runtimes are managed using Kubernetes CRI-O (all OCP deployments use CRI-O by default)

What are the Red Hat OpenShift Container Platform dependencies?

API Connect 10.0.7.0 can be deployed with the following versions of OpenShift Container Platform (OCP):

- version 4.12: supported on Red Hat Enterprise Linux 8.6; on-going certification for FIPS 140-2 and 140-3
- version 4.14: supported on Red Hat Enterprise Linux 8.6, 8.7, and 8.8; on-going certification for FIPS 140-2 and 140-3

For more information about OpenShift support for FIPS, see

- Current status on FIPS certification: Compliance Activities and Government Standards in the Red Hat Customer Portal

Install and configure OpenShift Container Platform

Configure settings for FIPS and IPSec, install the OpenShift cluster, and then configure the cluster to enable encryption and provide storage.

Procedure

1. Configure settings in the install-config.yaml file.
   Edit the install-config.yaml and make the following changes:
   - Set `fips: true`
   - Set the `networkType` to `OVNKubernetes`

   For example:

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OVNKubernetes
  serviceNetwork:
```

```
  - 172.30.0.0/16
fips: true
```

For an example of the install-config.yaml file, see Sample install-config.yaml file for bare metal in the Red Hat OpenShift documentation.

2. Create the IPSec installation manifests to allow IPSec tunneling for node-to-node communication in the OpenShift cluster.
   a. Generate the manifests from `install-config.yaml` by running the following command:

   ```
   openshift-install create manifests
   ```

   b. Create a cluster-network-03-config.yaml file with the following content, and save it in the `manifests` directory:

   ```
   apiVersion: operator.openshift.io/v1
   kind: Network
   metadata:
     name: cluster
   spec:
     defaultNetwork:
       type: OVNKubernetes
       ovnKubernetesConfig:
         ipsecConfig: {}
         mtu: 1400
   ```

3. Deploy the OpenShift cluster.
   Install the OpenShift cluster, and then install the OpenShift CLI.
   a. Install the OpenShift cluster as explained in Chapter 2. Installing the system in FIPS mode in the Red Hat OpenShift "FIPS support" documentation.
   b. Install the OpenShift CLI (oc) as explained in Installing the OpenShift CLI in the Red Hat OpenShift documentation.
4. Log in to the cluster as a user with `cluster-admin` privileges.

5. Validate the OpenShift cluster.
   a. Verify that FIPS is enabled on the OpenShift cluster.
      i. Run the following command to verify the `fips` setting:

      ```
      oc get cm cluster-config-v1 -n kube-system -o json | jq -r '.data."install-config"' | grep -i "fips"
      ```

      Review the results and verify that the `fips` setting is set to `true`:

      ```
      fips: true
        name: fips-apic
        root@fips-apic
      ```

      ii. Run the following command to get the names of the machine configurations:

      ```
      oc get mcp
      ```

      In the result, the configuration name appears in the `CONFIG` column; for example:

      ```
      NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED   MACHINECOUNT
      READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
      master   rendered-master-41831cf684b6e570518728abd24b9006   True      False      False      3              3
      3                   0                     15h
      worker   rendered-worker-2a4937983238e94a74307abc3eed2080   True      False      False      7              7
      7                   0                     15h
      ```

      For each configuration listed in the result, run the following command to verify that `Fips` is set to `true`:

      ```
      oc describe mc <CONFIG> | grep Fips
      ```

      The previous example returned two configurations, so run the command on each configuration:

      - `oc describe mc rendered-master-41831cf684b6e570518728abd24b9006 | grep Fips`
        `Fips:  true`
      - `oc describe mc rendered-worker-2a4937983238e94a74307abc3eed2080 | grep Fips`
        `Fips:  true`

   b. Verify that IPSec is enabled on the OpenShift cluster.
      i. Check the `ovn-ipsec` daemonset that manages the daemons responsible for configuring IPSec:

      ```
      oc get ds -n openshift-ovn-kubernetes ovn-ipsec
      ```

      A successful result looks like the following example:

      ```
      NAME        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR                 AGE
      ovn-ipsec   66666           beta.kubernetes.io/os=linux   3d3h
      ```

      ii. Verify that IPSec pods are running in all of the nodes in your OpenShift cluster:

      ```
      oc get pod -n openshift-ovn-kubernetes -o wide | grep ipsec
      ```

      A successful result looks like the following example:

      ```
       ovn-ipsec-4qp86        1/1     Running   0          38m   192.168.7.23   master2.ocp4.rober.lab   <none>
      <none>
       ovn-ipsec-pk7vh        1/1     Running   0          38m   192.168.7.21   master0.ocp4.rober.lab   <none>
      <none>
       ovn-ipsec-q4mwj        1/1     Running   0          22m   192.168.7.11   worker0.ocp4.rober.lab   <none>
      <none>
       ovn-ipsec-trz5m        1/1     Running   0          22m   192.168.7.12   worker1.ocp4.rober.lab   <none>
      <none>
       ovn-ipsec-vjmw8        1/1     Running   0          38m   192.168.7.22   master1.ocp4.rober.lab   <none>
      <none>
      ```

6. Enable `etcd` encryption with `aescbc` as explained in Encrypting etcd data in the Red Hat OpenShift documentation.

7. For local storage, choose either RHEL-provided disk encryption or Container Native Storage that uses RHEL-provided disk encryption.
   By storing all data in volumes that use RHEL-provided disk encryption and enabling FIPS mode for your cluster, both data at rest and data in motion (network data) are protected by FIPS-validated or Modules In Process encryption. You can configure your cluster to encrypt the root filesystem of each node, as described in Customizing nodes in the Red Hat OpenShift documentation.

Install API Connect

Install API Connect on the OpenShift cluster, and ensure that server connections support FIPS.

Procedure

1. Install API Connect.
   For instructions on installing API Connect with the top-level CR, see Installing with the top-level CR on OpenShift in this documentation. For instructions on installing API Connect with subsystem CRs, see Installing API Connect in a single namespace on OpenShift or Installing API Connect subsystems in different namespaces or environments on OpenShift in this documentation.

2. Ensure that all server connections support FIPS.

   Inbound connections: ensure that the Gateway API invocation endpoint supports FIPS
   The following list shows the inbound connections to an API Connect deployment:

```
$ oc get routes
NAME                        HOST/PORT                                                    PATH   SERVICES
PORT   TERMINATION           WILDCARD
apic-a7s-ai-endpoint        apic-a7s-ai-endpoint-apic.apps.fips-apic.cp.fyre.ibm.com             apic-a7s-mtls-gw
4443   passthrough           None
apic-gw-gateway             apic-gw-gateway-apic.apps.fips-apic.cp.fyre.ibm.com                  apic-gw-datapower
9443   passthrough           None
apic-gw-gateway-manager     apic-gw-gateway-manager-apic.apps.fips-apic.cp.fyre.ibm.com          apic-gw-datapower
3000   passthrough           None
apic-mgmt-admin             apic-mgmt-admin-apic.apps.fips-apic.cp.fyre.ibm.com                  apic-mgmt-juhu
2005   reencrypt/Redirect    None
apic-mgmt-api-manager       apic-mgmt-api-manager-apic.apps.fips-apic.cp.fyre.ibm.com            apic-mgmt-juhu
2006   reencrypt/Redirect    None
apic-mgmt-consumer-api      apic-mgmt-consumer-api-apic.apps.fips-apic.cp.fyre.ibm.com           apic-mgmt-juhu
2001   reencrypt/Redirect    None
apic-mgmt-platform-api      apic-mgmt-platform-api-apic.apps.fips-apic.cp.fyre.ibm.com           apic-mgmt-juhu
2000   reencrypt/Redirect    None
apic-ptl-portal-director    apic-ptl-portal-director-apic.apps.fips-apic.cp.fyre.ibm.com         apic-ptl-nginx
4443   passthrough           None
apic-ptl-portal-web         apic-ptl-portal-web-apic.apps.fips-apic.cp.fyre.ibm.com              apic-ptl-b1aab746-
www   4443   reencrypt/Redirect   None
```

   All of the inbound connections support FIPS by default, except for the Gateway API invocation endpoint.

   The TLS server profile used by the DataPower Gateway for API invocation is configured from the Cloud Manager when you create a gateway service. To be FIPS compliant, the TLS server profile used with a gateway service must be configured with FIPS-only ciphers to ensure that the server only accepts FIPS connections.

   To configure the TLS server profile, use the Cloud Manager's Register Service > DataPower Gateway page, which you can access from the Topology menu. Locate the API invocation endpoint section and use the Server Name Indication (SNI) - TLS server profile setting to create the server profile. Use a TLS server profile that is FIPS-compliant (for example, do not use: **`TLS_CHACHA20_POLY1305_SHA256`**, which is not compliant). For information on configuring the TLS server profile, see Registering a gateway service in this documentation.

   Outbound connections: Ensure that all of the outbound connections support FIPS
   For outbound connections, it is your responsibility to ensure that the external servers configured with API Connect accept only FIPS connections. The following list shows the typical outbound connections for an API Connect deployment
   - OIDC providers (Google, Facebook, Keycloak etc.)
   - LDAP providers
   - Authentication URLs
   - External user registry
   - Target URLs used in APIs
   - Remote backups
   - Auditing endpoints
   - SMTP endpoints
   - Analytics offloading
   - Analytics backup to S3 storage
   - Automated API behavior testing tool (ATM) - calls out to the APIs being tested
   - Portal - Potential drupal extensions installed by customer
   - Portal - SFTP and S3 backups
   - 2-data-center deployments - Calls to other clusters

Preparing for installation

Set up your OpenShift environment in preparation for deploying API Connect.

About this task

Ensure that your environment meets all prerequisites, plan your deployment, and download files for installation.

Procedure

1. Ensure that your environment meets the following prerequisites:
   - IBM API Connect Version 10 software product compatibility requirements
   - Deployment requirements
   - If you want to use the OpenShift web console, Cluster admin access is required.
   - API Connect requires `block storage` as the storage class for all subsystems.
   - Set the default SCC (security context constraint) to `restricted`.
     API Connect supports only the `restricted` SCC on OpenShift.
   - If your OpenShift deployment uses a LimitRange and a CPU `maxLimitRequestRatio` is set, then its value must be at least 25; otherwise some API Connect pods might fail to start.
   - S3 object-store available for storing API Connect subsystem database backups.
2. Plan your deployment as explained in Planning your deployment.
3. Review Operator, CASE, and platform requirements to determine the operator, operand, and CASE versions that you need for your deployment.
4. Optional: Specify additional settings in each subsystem CR to define the deployment's configuration before beginning the installation.
   In particular, consider increasing the timeout value for Management endpoints as explained in Configuring timeouts for management endpoints.

   For information on other configuration settings, see API Connect configuration settings.

   Tip: If you choose not to specify configuration settings now, you can modify the installation CRs later and then re-apply them to update the deployment.

- **Operator, CASE, and platform requirements**
  Before you install API Connect on Red hat OpenShift Container Platform, verify that you are using the correct operator, CASE file (for air-gapped installations), and OpenShift.
- **API Connect configuration settings**
  Specify configuration settings for API Connect in the CR, or the Platform UI in Cloud Pak for Integration.
- **Configuring timeouts for management endpoints**
  Adjust the timeout settings for management endpoints by updating the CR.

Operator, CASE, and platform requirements

Before you install API Connect on Red hat OpenShift Container Platform, verify that you are using the correct operator, CASE file (for air-gapped installations), and OpenShift.

Table 1 lists the operator, operand, and CASE information that applies to each release of API Connect.

Supported versions of operators for API Connect

Table 1 lists the operators that are used for installing and upgrading API Connect.

Table 1. Operators for installing and upgrading API Connect

API Connect version	Data Power version	API Connect Operator channel	API Connect Operator version	DataPower Operator version	Foundational services channel version
10.0.7.0	10.5.3.0	5.1	5.1.0	1.9.0	4.3 to 4.6
10.0.6.0	10.5.1.0	5.0	5.0.0	1.7.0	3.23

CASE versions required for air-gapped installation and upgrade

Table 2 lists the required CASE file for each API Connect release:

Table 2. CASE versions for air-gapped deployments

API Connect version	API Connect CASE version	Foundational services CASE version	EDB CASE version
10.0.7.0	5.0.1	5.0.0	4.18.0
10.0.6.0	5.0.0	1.15.10	(none)

Supported versions of Red Hat OpenShift Container Platform

Table 3 provides a quick reference of the supported OpenShift versions for each API Connect release:

Table 3. API Connect and OpenShift Container Platform (OCP) compatibility

API Connect version	OCP 4.10	OCP 4.12	OCP 4.14
10.0.5.3	✓	✓	
10.0.5.4	✓	✓	
10.0.5.5	✓	✓	✓
10.0.6.0	✓	✓	
10.0.7.0		✓	✓

API Connect configuration settings

Specify configuration settings for API Connect in the CR, or the Platform UI in Cloud Pak for Integration.

You can use the Platform UI on Cloud Pak for Integration, the OpenShift web console, or direct editing of the top-level or subsystem CR YAML files to specify advanced configuration options for API Connect. The following topics describe the available configuration options.

Attention: A mistake in a configuration update can result in an admission webhook error when `APIConnectCluster` is updated, or the `APIConnectCluster` can get stuck in a pending state with a webhook error message.

- Example of an admission webhook error when the `APIConnectCluster` update is rejected:
  In this example, the user attempted to change the deployment profile to `n12xc4.m12-incorrect` and the result looked like the following example:

  ```
  oc edit apiconnectcluster

  error: apiconnectclusters.apiconnect.ibm.com "production" could not be patched: admission webhook
  "vapiconnectcluster.kb.io" denied the request: APIConnectCluster.apiconnect.ibm.com "production" is invalid: spec.Profile:
  Invalid value: "n12xc4.m12-incorrect": invalid profile. Options are: [n3xc4.m16 n12xc4.m12]
  ```

- Example of when the `APIConnectCluster` update is allowed, but is the state changes pending due to an underlying webhook error:

  ```
  oc get apiconnectcluster -n apic

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

Basic configuration

- Basic configuration settings

Advanced configuration

To enable access to the advanced configuration settings with Platform UI, toggle the Advanced Options slider.

- Analytics subsystem settings
- Gateway subsystem settings
- Management subsystem settings
- Developer Portal subsystem settings
- Additional Kubernetes settings

Additional configuration

You must edit the yaml file to set additional properties.

- **Basic configuration settings**
  You can specify the API Connect basic configuration settings directly in the CR or with the Platform UI in Cloud Pak for Integration.
- **Management subsystem settings**
  You can specify advanced configuration settings for the Management subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- **Gateway subsystem settings**
  You can specify advanced configuration settings for the Gateway subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- **Developer Portal subsystem settings**
  You can specify advanced configuration options for the Developer Portal subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.
- **Analytics subsystem settings**
  Specify advanced configuration settings for the analytics subsystem directly in the analytics CR. If you are using Cloud Pak for Integration, you can configure many of the analytics settings in Platform UI.
- **Additional Kubernetes settings**
  For each API Connect subsystem, you can specify additional settings for annotations and pod management.

Basic configuration settings

You can specify the API Connect basic configuration settings directly in the CR or with the Platform UI in Cloud Pak for Integration.

Review the tables for information on configuration settings for the following areas:

- Details
- License
- Deployment profiles for installing with a single, top-level CR
- Product version
- Storage class

Details

Table 1. Details

Field	Type	Required	Description
Name	String	Yes	Name this instance using only lowercase alphanumeric characters plus the underscore "-". Maximum of 54 characters. Default: none

License

Table 2. License

Field	Type	Required	Description
License acceptance	Boolean	Yes	The license agreement https://ibm.biz/apictoolkitlic must be accepted before you can install the API Connect capability. Default value: **`false`** (off). In the UI, click the License acceptance toggle to enable it. In the CR, set the **`accept:`** field to **`true`** to accept the license.
License use	String	Yes	Select **`production`** or **`nonproduction`**, to match the license you purchased. Default: **`nonproduction`**
License metric	Enumerated	No	Enter the unit of measure that is used for your program license: OpenShift • PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used. • MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program. Cloud Pak for Integration • VIRTUAL_PROCESSOR_CORE - Default value. If you leave the field blank, this value is used. • MONTHLY_API_CALL - Applies only to the IBM Cloud Pak for Integration - API Calls add-on program. For information on tracking monthly call volume, see Tracking API volume for auditing and compliance.
License ID	String	Yes	Select the license that you purchased for this instance of API Connect. Valid licenses are based on the selected value in the Product version field (see Table 4). When deploying the standalone IBM API Connect product, you must select the License ID for the API Connect program that you purchased. To view all License IDs, see API Connect licenses.

Deployment profiles for installing with a single, top-level CR

Select a deployment profile for the new installation. See API Connect deployment profiles for OpenShift and Cloud Pak for Integration for more information on the available profiles.

For information on switching to a different deployment profile after installation, see the following topics:

- Changing the deployment profile on Cloud Pak for Integration
- Changing deployment profiles on OpenShift top-level CR

Product version

Table 3. Product version

Field	Type	Required	Description
Product version	Enumerated	No	Select the product version to be installed in the system. Warning: Do not select a channel here instead of the API Connect version. Selecting a channel will result in any API Connect interim fix updates being applied automatically and without warning.

Storage class

Table 4. Storage class

Field	Type	Required	Description
Default Storage Class	Enumerated	No	Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is optional, but is highly recommended. If not specified the default cluster storage class is used. See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pack release level: https://www.ibm.com/support/knowledgecenter/SSGT7J. In the navigation panel, select System Requirements > Storage. To review API Connect storage support, see Storage classes.

Management subsystem settings

You can specify advanced configuration settings for the Management subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

For annotations, labels, node selectors, and tolerations, see Additional Kubernetes settings

Note: Optionally, you might want to increase the timeout settings for management endpoints, particularly for large deployments. See Configuring timeouts for management endpoints.

Database backup settings

The settings in this section apply to API Connect V10.0.7.0 and later. For v10.0.6.0, see Database backup settings V10.0.6.0.

Table 1. Database backup

Field	Type	Required	Description
Credentials	Drop down menu	Yes	The name of the Kubernetes secret containing your S3 Access Key / Key Secret. Default: (Generated Value) See Configuring management database backups.
Server hostname	String	Yes	The backups host. Specify S3 endpoint with the corresponding S3 region in the format **<S3endpoint>/<S3region>**. For example: **s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard** Default: none See Configuring management database backups.
Path	String	Yes	The path to the location of the backup. The name of your S3 bucket to store backup data. For example, **bucket-name/folder**. The use of subdirectories in the bucket name is not supported. Ensure that **bucket-name/folder** is empty. Default: none See Configuring management database backups.
Schedule	String	Yes	Cron like schedule for performing automatic backups. The format for the schedule is: • ****** • ------ • |||||| • ||||| +----- day of week (0 - 6) (Sunday=0) • |||| +------- month (1 - 12) • ||| +--------- day of month (1 - 31) • || +----------- hour (0 - 23) •	+------------- min (0 - 59) • +--------------- sec (0 - 59) The backup schedule defaults to **0 0 * * * ***. This means a backup is run every day at midnight and minute zero UTC. When you configure a host, if you do not specify a value for **schedule**, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed. See Configuring management database backups.
Backup certs	Drop down menu	Optional, if S3 object-store requires them.	Secret containing the CA certificate for your S3 object-store. Required for API Connect ot trust the S3 object-store.
RepoRetentionFull	Integer	Yes	The number of days to retain backups for. For example, if **RepoRetentionFull** is set to 30, then any backups older than 30 days are deleted.
Backup S3 URI Style	Drop down menu	**host**	Indicates whether URIs to your S3 backup specify a host or a path value. Set to either **host** or **path**.

Database backup settings V10.0.6.0

Table 2. Database backup v10.0.6.0

Field	Type	Required	Description
Credentials	Drop down menu	No. Optional	• For **objstore** type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret • For **SFTP** type backups, the name of the Kubernetes secret containing your SFTP Username/password Default: (Generated Value) See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
Server hostname	String	No. Optional	The backups host: • For **objstore** type backup, specify S3 endpoint with the corresponding S3 region in the format **<S3endpoint>/<S3region>**. For example: **s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard** • For **sftp** type, the SFTP server hostname. For example: **203.0.113.0** or **example.com** • Not required for local storage backups Default: none See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.

Field	Type	Required	Description
Path	String	No. Optional	The path to the location of the backup: • For **objstore** backups, the name of your S3 bucket to store backup data. For example, **bucket-name/folder**. The use of subdirectories in the bucket name is not supported. Ensure that **bucket-name/folder** is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. • For **sftp** type, the folder name on the SFTP server. For example: **mgmt_backup**. • Not required for local storage backups Default: none See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
Server port	String	No. Optional	Backup server port The port for the protocol to connect to the **host**. The backup port is not required for object storage. Default: **22** See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
Protocol	String	No	The type of the backup. Supported types: • **objstore** • **sftp** The default value is **sftp**. • Local storage backups For local backups, do not specify this parameter. When the parameter is set to "", the backup type is set to local storage backup. See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
Restart database	Toggle Switch	No. Optional	Flag to accept database reconfiguration. Restarting the database is required when changing the backup configuration. When marked true and the new backup configuration differs, the operator will stop the database and any dependent services, apply the change, and restart the stopped components. Defaults to Off. For more information on restarting the database, see Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0
Retries	String	No. Optional	For **sftp** type backups only, the number of times the **ibm-apiconnect** Operator attempts backups in the event of a failed SFTP backup. Not used for **objstore** or local storage backup. Default value: **0**. See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
S3 provider type	Drop down menu	Required when using s3 provider.	Name of the S3 provider to use. Not required for local backups or SFTP backups. If you are configuring backups with an **s3provider**, you must specify one of the supported values of **aws** or **ibm**. See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
Schedule	String	No. Optional	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The backup schedule defaults to **0 0 * * ***. This means a backup is run every day at midnight and minute zero UTC. When you configure a host, if you do not specify a value for **schedule**, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed. See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.

Database persistence

Table 3. Database persistence

Field	Type	Required	Description
Storage Class	Drop down menu	No. Optional	Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is highly recommended. If not specified the default cluster storage class is used. See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pack release level: https://www.ibm.com/support/knowledgecenter/SSGT7J. In the navigation panel of the KC, select System Requirements > Storage. To review API Connect storage support, see Storage classes.

Field	Type	Required	Description
Volume size	Integer	No. Optional	Represents the size of the Database volume. Defaults to 120G for `n1xc4.m16` installation profile, 180G for `n3xc4.m16` installation profile.

Site name

Table 4. Site Name

Field	Type	Required	Description
Site name	string	No. Optional	Site name is the logical name of this failure domain. Typically site name is the name of your data center, such as Amazon AZ. Default: randomly generated name. This will be the site name of the management deployment. For example, given the site name of `test` your management deployment will look like `management-test-postgres`.

Test and Monitor

Table 5. Test and Monitor

Field	Type	Required	Description
Enable Test and Monitor	Boolean. Toggle off/on	No. Optional	Test and Monitor automatically generates API test assertions. It contains dashboards for monitoring API test assertions and getting alerts on the health of your APIs. Default: off See Installing the Automated API behavior testing application.

Billing

Table 6. Billing

Field	Type	Required	Description
Enable Billing	Boolean. Toggle off/on	No. Optional	To enable the monetization of your Product Plans, you must add a billing integration resource in your API Connect provider organization that defines the configuration data needed to synchronize with an external subscription billing system. Default: off See Adding a billing integration resource.

Gateway subsystem settings

You can specify advanced configuration settings for the Gateway subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

- v5 compatibility mode
- Operations Dashboard
- Token management service
- Autoscaling gateway pods

For annotations, labels, node selectors, and tolerances, see Additional Kubernetes settings

v5 compatibility mode

Table 1. v5 compatibility mode

Field	Type	Required	Description
V5 comparability mode	Toggle Switch off/on	No. Optional	Enabling this provides v5 compatibility (trading off some performance) with the IBM DataPower Gateway that was provided with IBM API Connect Version 5 and earlier releases. Consider using this DataPower Gateway (v5 compatible) if you are an existing DataPower user and want to utilize your DataPower resources and knowledge. Default: off To review v5 compatibility, see API Connect gateway types.

Operations Dashboard

Table 2. Operations Dashboard

Field	Type	Required	Description

Field	Type	Required	Description
Enable Operations Dashboard tracing	Toggle Switch off/on	No. Optional	Enable or disable cross-component transaction tracing to allow troubleshooting and investigating errors and latency issues. Default: disabled (off) To review the Operations Dashboard, see https://www.ibm.com/support/knowledgecenter/SSGT7J_20.3/tracing/operations_dashboard.html For information on configuring the Operations Dashboard, see https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta2/#odtracing.
Operations Dashboard namespace	string	No. Optional	Namespace where the IBM Cloud Pak for Integration Operations Dashboard has been deployed. Default: none
Replica count	string	No. Optional	Specify the number of instances of the Gateway that should be deployed. For high-availability, a minimum of 3 replicas (one per node) is recommended. Default: none

Token management service

Table 3. Token management service

Field	Type	Required	Description
Advanced: Enable TMS	Toggle Switch	No. Optional	Enable or disable token management service (TMS) to use the DataPower to manage the token lifecycle.. Default: disabled (off) The Token Management Service uses the DataPower distributed cache to manage the token lifecycle which includes when to revoke access rights. For more information, see Token management with DataPower API Gateway.
Advanced: Storage Class	string	No. Optional	Specify the RWO block storage class to use for persistence storage. This will be used when creating a PVC. Specification of the storage class is highly recommended. If not specified the default cluster storage class is used. See the IBM Cloud Pak documentation for guidance for Storage class selection. Go to the following page and select your Cloud Pack release level: https://www.ibm.com/support/knowledgecenter/SSGT7J. In the navigation panel of the KC, select System Requirements > Storage. To review DataPower storage configuration, see https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta2/#storage.
Advanced: Volume size	string	No. Optional	Resources represents the minimum resources the volume should have. For example, 10Gi.Default: none To review DataPower storage configuration, see **storage** API. in the DataPower documentation.

Autoscaling gateway pods

Autoscaling enables the gateway deployment to scale dynamically, either vertically or horizontally. It is not possible to use both horizontal and vertical pod scaling simultaneously. You must configure either **HPA** (for horizontal scaling) or **VPA** (for vertical scaling) as the scaling method.

Note: DataPower Gateway supports long-lived connections such as such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when pods are scaled down based on defined CPU or memory thresholds. Workloads with long-lived connections are more vulnerable to failed API transactions when auto-scaling is occurring.
To learn more about HPA and VPA autoscaling, see the following topics in the DataPower documentation:

- Autoscaling DataPower Pods
- **podAutoScaling** API

You can configure autoscaling for Gateway pods by inserting a new subsection in the **spec** **gateway** section of the top-level CR, or by filling in the corresponding fields in the OpenShift or IBM Cloud Pak Platform UI. Settings are described in Table 4.

The following example configures HPA autoscaling in the gateway section of the top-level API Connect CR:

```
spec:
  gateway:
    podAutoScaling:
      hpa:
        maxReplicas: 5
        minReplicas: 3
        targetCPUUtilizationPercentage: 90
        targetMemoryUtilizationPercentage: 80
      method: HPA
```

Table 4. Autoscaling configuration settings

UI Field name / CR Setting	Autoscaling method	Required	Description
• Scaling method • **method**	All	Yes	Select the autoscaling method to be used: either **HPA** (Horizontal Pod Autoscaling) or **VPA** (Vertical Pod Autoscaling). To disable autoscaling, leave **method** blank, or omit the **podAutoScaling:** section from the CR.
• Minimum pods • **minReplicas**	HPA	Yes	Minimum number of pods that can be scaled down to by the horizontal pod autoscaler.
• Maximum pods • **maxReplicas**	HPA	Yes	Maximum number of pods that can be scaled up to by the horizontal pod autoscaler. Must be a value greater than **minReplicas**.

UI Field name / CR Setting	Autoscaling method	Required	Description
• Target CPU Utilization Percentage • `targetCPUUtilizationPercentage:`	HPA	No	Percent threshold on average CPU usage by pods for triggering horizontal scaling.
• Target Memory Utilization Percentage • `targetMemoryUtilizationPercentage`	HPA	No	Percent threshold on average memory usage by pods for triggering horizontal scaling.
• Maximum CPU • `maxAllowedCPU`	VPA	No	Maximum value for CPU resource that can be assigned to a DataPower container when scaling vertically. Minimum value is set from `DataPowerRequestsSpec.ResourceCPU`
• Maximum Memory • `maxAllowedMemory`	VPA	No	Maximum amount of memory that can be assigned to a DataPower container when scaling vertically. Minimum value is set from `DataPowerRequestsSpec.ResourceMemory`

Developer Portal subsystem settings

You can specify advanced configuration options for the Developer Portal subsystem directly in the CR or with the Platform UI in Cloud Pak for Integration.

The IBM Cloud Pak Platform UI enables you to specify settings for the following areas:

- Database backups
- Site name

For annotations, labels, node selectors, and tolerations, see Additional Kubernetes settings

Known limitation: When deployed on OpenShift or Cloud Pak for Integration, the API Connect Developer Portal subsystem does not support multiple hosts for the `portalUIEndpoint` endpoint, due to a limitation of Routes in OCP.

Database backups

Table 1. Database backup settings

Field	Type	Description
Credentials	String	For `objstore`, this is the name of your Kubernetes secret that was generated with your username and password for your backup database. For `sftp`, this is the secret that contains the credentials for your `sftp` username and password. Required if configuring backups. See also Configuring portal database backups.
Backup host	String	The S3 endpoint with the corresponding S3 region in the format `<S3endpoint>/<S3region>`. For example: `s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard` For sftp, this is the sftp server host name. Required if configuring backups. See also Configuring portal database backups.
Backup path	String	For `objstore`, the name of your S3 bucket to store backup data. For example: `test-bucket/restore-test`. For sftp, this is the path to the folder on the sftp server. Required if configuring backups. See also Configuring portal database backups.
Backup port	Integer	This is the port for the protocol to connect to the host. There is no default. See also Configuring portal database backups.
Backup protocol	String	Choose the protocol (sftp or objstore) you wish to use to communicate with your remote backup endpoint. Objstore is recommended for most reliability. There is no default. This field must be set to either `sftp` or `objstore` to configure backups. See also Configuring portal database backups.
Backup schedule	String	This is the schedule in cron string format for how often automatic Developer Portal backups are run. For example: `0 /3 * * *` means at minute 0 past every 3rd hour. There is no default. Required if configuring backups. See also Configuring portal database backups.

Site name

Table 2. Site name

Field	Type	Description
Site name	string	Site name is the logical name of the failure domain of this portal subsystem. Typically site name is the name of your data center, such as Amazon AZ. The default is a randomly generated name.

Analytics subsystem settings

Specify advanced configuration settings for the analytics subsystem directly in the analytics CR. If you are using Cloud Pak for Integration, you can configure many of the analytics settings in Platform UI.

Review Analytics preinstallation planning to decide on the deployment options for your analytics subsystem. If you want to offload your analytics data and disable local storage, you must configure local storage disablement before installation because it cannot be changed after installation.

For annotations, labels, node selectors, and tolerations, see Additional Kubernetes settings

Cloud Pak for Integration Platform UI

In the Cloud Pak for Integration Platform UI you can specify settings for:

- Analytics database backup
- Storage type, class, and size

All other analytics settings must be configured directly in the analytics section of the `APIConnectCluster` CR.

Table 1. Database backup

Field	Type	Required	Description
Backup chunk size in GB	Integer	No	`chunkSize` specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB. Default value is 1 GB. See Configuring analytics backups.
Credentials	String	No	The backup credential is the name of the Kubernetes secret that you created for accessing your S3-compatible storage. No default. See Configuring analytics backups.
Compress	Boolean: off or on	No	Determines whether metadata files are stored in compressed format. Default value is false (off). See Configuring analytics backups.
Enable server-side encryption	Boolean: off or on	No	Determines whether files are encrypted. When set to true (on), files are encrypted on the server side by using AES256. Default is false (off). See Configuring analytics backups.
Analytics server hostname	String	No	The URL to your S3 storage. No default. See Configuring analytics backups.
Backup Path	String	No	The path to your S3 storage at the URL specified by analytics server hostname.host. No default. See Configuring analytics backups.
Schedule cron expression	String	No	The schedule that determines how often backups are run automatically. The format for the schedule is any valid `cron` string. Default: "*/20 * * * *" See Configuring analytics backups.

Table 2. Storage type, class, and size settings

Field	Type	Required	Description
Master storage class	String	Yes	If you specified dedicated storage, the master storage class is the storage class that is used by your `storage-os-master` pods. Select from the drop down. Local-volume storage is recommended for best performance. If none selected, the default storage class of your cluster is used. See Analytics preinstallation planning.
Shared storage class	String	Yes	The storage class used by your analytics `storage` pods. Select from the drop down. Local-volume storage is recommended for best performance. If none selected, the default storage class of your cluster is used. See Analytics preinstallation planning.
Master volume size	Integer > 0	No	If you specified dedicated storage, then master volume size specifies the disk storage that is allocated to your `storage-os-master` pods. The `storage-os-master` pods do not store your analytics data, and if left unset the value defaults to 5Gi.
Shared volume size	Integer > 0	Yes	How much persistent disk storage is allocated for analytics data (in Gi). See Estimating storage requirements. If you do not have the information to estimate your usage, set this to 500Gi. For an installation where you expect to store little or no analytics data, you can set the storage to 50Gi.
Type	Boolean: dedicated or shared	Yes	OpenSearch storage type to use for your analytics data. For more information about storage types, see Analytics preinstallation planning. The default storage type is shared.

Additional Kubernetes settings

For each API Connect subsystem, you can specify additional settings for annotations and pod management.

The additional settings are generic Kubernetes configuration settings. The Platform UI advanced configuration option for API Connect enables you to set them specifically for each subsystem.

Annotations

The annotations field serves as a pass-through for all the resources managed by the CR. You can add any annotation to this field. The annotations here overwrite the default annotations.

See https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/.

Labels

The labels field serves as a pass-through for Pod labels. You can add any label to this field and have it apply to the pod. The labels here overwrite the default labels provided. For example, if the label 'mylabel=1' was provided, it would apply to all kubernetes resources managed by this CR.

See https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/.

Node Selector

This field allows you to specify a set of key value pair that must be matched against the node labels to decide which API Connect pods can be scheduled on that node. Only nodes matching all of these key value pairs in their labels will be selected for scheduling API Connect pods.

See https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/.

Tolerations

This field allows you to select which nodes API Connect pods can be scheduled or not scheduled. By specifying a set of matching tolerations, you can control whether specific nodes can be used for scheduling API Connect Pods.

Effect (optional)
: Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

Key (optional)
: Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

  Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

Operator (optional)
: Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

Toleration seconds (optional)
: TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

Value (optional)
: Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

See https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/.

Configuring timeouts for management endpoints

Adjust the timeout settings for management endpoints by updating the CR.

About this task

On large deployments, or with large API usage, you might encounter timeout issues when deployed with the default timeout setting of 30 seconds. To avoid timeout issues, increase the setting for the management endpoints used in routing.

You do not have to set these values on initial installation. You can set them or change them after installation and reapply the CR YAML files.

Procedure

Edit the CR to set the timeout to 240 seconds.
The instructions for editing depend on whether you install API Connect using subsystem CRs or a single top-level CR:

- If you are installing API Connect using subsystem CRs, edit **`management_CR.yaml`**:

```
  cloudManagerEndpoint:
    annotations:
      haproxy.router.openshift.io/timeout: 240s
    hosts:
    - name: admin.example.com
      secretName: cm-endpoint
  apiManagerEndpoint:
    annotations:
      haproxy.router.openshift.io/timeout: 240s
    hosts:
    - name: manager.example.com
      secretName: apim-endpoint
  platformAPIEndpoint:
    annotations:
      haproxy.router.openshift.io/timeout: 240s
    hosts:
    - name: api.example.com
```

```
      secretName: api-endpoint
  consumerAPIEndpoint:
    annotations:
      haproxy.router.openshift.io/timeout: 240s
    hosts:
    - name: consumer.example.com
      secretName: consumer-endpoint
```

- If you are installing API Connect using a top-level CR, the management subsystem settings are embedded in the top-level APIConnectCluster CR. Edit the endpoint settings under `spec:management:` as follows:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  name: <name_of_your_instance>
  namespace: ${NAMESPACE}
spec:
...
  management:
    cloudManagerEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
      - name: admin.example.com
        secretName: cm-endpoint
    apiManagerEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
      - name: manager.example.com
        secretName: apim-endpoint
    platformAPIEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
      - name: api.example.com
        secretName: api-endpoint
    consumerAPIEndpoint:
      annotations:
        haproxy.router.openshift.io/timeout: 240s
      hosts:
      - name: consumer.example.com
        secretName: consumer-endpoint
...
```

Installation procedures with IBM Cloud Pak for Integration

The API Management capability of IBM Cloud Pak for Integration uses IBM API Connect.

There are several methods for installing API Connect with Cloud Pak for Integration:

- Use the Cloud Pak for Integration Platform UI to install API Connect with a top-level CR. The steps are documented in the installation section of the Cloud Pak for Integration documentation: https://www.ibm.com/docs/en/cloud-paks/cp-integration.
- If API Connect is already installed with the top-level CR and you want to migrate it to Cloud Pak for Integration, see Migrating from OpenShift to Cloud Pak for Integration on the same cluster.

Installing API Connect in a single namespace on OpenShift

Use individual subsystem CRs to install API Connect subsystems in a shared namespace on OpenShift.

About this task

API Connect subsystems are installed by creating individual subsystem CRs in the same shared namespace. If you want to install your API Connect subsystems in different namespaces, for example Management server in a namespace that is called **`mgmt`**, and gateway in a namespace that is called **`gway`**, then see: Installing API Connect subsystems in different namespaces or environments on OpenShift.

Air-gapped installations: If you are installing API Connect in an offline environment, first follow the steps for configuring the catalog sources as described here: Air-gapped installation.

Attention: If you want to configure FIPS support for API Connect, you must enable FIPS on the cluster as part of the initial OpenShift deployment. You cannot modify an existing cluster to enable FIPS support. For more information, see Configuring FIPS support on OpenShift.

1. Installing operators
   Install operators for API Connect, DataPower, and common services so that you can deploy the API Connect subsystems in a shared namespace on OpenShift.
2. Setting up a certificate issuer
   Create the cert-manager issuer for generating the certificates and secrets for use with API Connect.
3. Installing the Management subsystem in a shared namespace
   Install the Management subsystem by creating and applying the mgmt_cr.yaml file.

4. Installing the Gateway subsystem in a shared namespace
   Install the Gateway subsystem by creating and applying the apigateway_cr.yaml file.
5. Installing the Portal subsystem in a shared namespace
   Install the Portal subsystem by creating and applying the portal_cr.yaml file.
6. Installing the Analytics subsystem in a shared namespace
   Install the analytics subsystem by creating and applying the analytics_cr.yaml file.

Installing operators

Install operators for API Connect, DataPower, and common services so that you can deploy the API Connect subsystems in a shared namespace on OpenShift.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.7.0	v5.1	5.1.0

Note: There can be only one instance of the API Connect operator in your OpenShift environment. You cannot have multiple namespaces each with their own API Connect operator.
You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

Procedure

1. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step 3.
   The **ibm-pak** plug-in enables you to access hosted product images, and to run **oc ibm-pak** commands against the cluster. To confirm that **ibm-pak** is installed, run the following command and verify that the response lists the command usage:

   ```
   oc ibm-pak --help
   ```

2. Obtain an entitlement key for the Entitled Registry:
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step 3.
   a. Log in to the IBM Container Library.
   b. In the Container software library, select Get entitlement key.
   c. After the Access your container software heading, click Copy key.
   d. Copy the key to a safe location.
3. Create the namespace where API Connect will be installed, either by selecting Home > Projects > Create Project in the UI, or with the following command:

   ```
   oc create ns <APIC-namespace>
   ```

   The namespace where you install API Connect must meet the following requirements:
   - Red Hat OpenShift: Only one top-level CR (**APIConnectCluster**) can be deployed in each namespace.
   - Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.
   - Red Hat OpenShift restricts the use of default namespaces for installing non-cluster services. The following namespaces cannot be used to install API Connect:
     - **default**
     - **kube-system**
     - **kube-public**
     - **openshift-node**
     - **openshift-infra**
     - **openshift**
4. If you are installing the operator into a single namespace, create an OperatorGroup object specifying that namespace.
   Attention: If you are installing the operator into all namespaces, skip this step because the **openshift-operators** namespace already has an appropriate Operator group in place.
   a. Create the apiconnect-operator-group.yaml file with an OperatorGroup object that identifies the namespace in which RBAC permissions for the IBM API Connect Operator will be generated, and where the CSV will be available:

      ```
      apiVersion: operators.coreos.com/v1
      kind: OperatorGroup
      metadata:
        name: <operator-group-name>
        namespace: <APIC-namespace>
      spec:
        targetNamespaces:
        - <APIC-namespace>
      ```

   b. Apply the OperatorGroup object with the following command:

      ```
      oc apply -f apiconnect-operator-group.yaml
      ```

5. If you are installing the operator in all namespaces, then create the common services namespace:

   ```
   oc create ns ibm-common-services
   ```

6. Create a pull secret in the namespaces where you want to install API Connect.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   a. Open the Red Hat OpenShift web console, click Workloads > Secrets.
   b. Ensure that the Project is set to the namespace where you intend to install API Connect.
   c. Click Create and select Image pull secret.

d. Set the following parameters for the secret:
- Set Secret name to **ibm-entitlement-key**.
- Set Authentication type to **Image registry credentials**.
- Set Registry server address to **cp.icr.io**.
- Set Username to **cp**.
- Set Password to the entitlement key generated in Step 2.
- Click Create to create the secret.

Note: If your Red Hat OpenShift platform is Red Hat OpenShift on IBM Cloud (ROKS), you might need to reload your worker nodes after the image pull secret is created. You can reload the worker nodes either from the Red Hat OpenShift web console or from the command line with the following command: **ibmcloud oc worker reload**
For information on using the **reload** command, see the Red Hat OpenShift on IBM Cloud documentation.

7. Add the API Connect catalog sources to your cluster.
Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
Adding catalog sources to your Red Hat OpenShift cluster adds the IBM operators to the list of operators you can install. For most connected and air-gapped clusters, applying individual catalog sources is the most effective way to fully control software versioning on the cluster.

   a. Generate the catalog source files:
      i. Log into your cluster using the **oc login** command and your user credentials:

      ```
      oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
      ```

      ii. Export the variables for the command line to use:

      ```
      export CASE_NAME=ibm-apiconnect
      export CASE_VERSION=5.1.0
      export ARCH=amd64
      ```

      For information on API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

      iii. Download the files for the operators required by API Connect:

      ```
      oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
      ```

      iv. Generate the catalog sources for API Connect:

      ```
      oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
      ```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources (run both commands):

      ```
      cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
      ```

      ```
      cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
      ```

   b. Apply the catalog sources:
      i. Apply the catalog sources (run both commands):

      ```
      oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
      ```

      ```
      oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
      ```

      ii. Confirm that the catalog sources have been created in the **openshift-marketplace** namespace:

      ```
      oc get catalogsource -n openshift-marketplace
      ```

8. Add the IBM Cloud Pak foundational services (common services) catalog sources to your cluster.
Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   a. Generate the catalog source files:
      i. Log into your cluster using the **oc login** command and your user credentials:

      ```
      oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
      ```

      ii. Export the variables for the command line to use; substituting the appropriate CASE version:

      ```
      export CASE_NAME=ibm-cp-common-services
      export CASE_VERSION=<case version>
      export ARCH=amd64
      ```

      Set *<case version>* to the value specified for *Foundational services channel version* in Operator, CASE, and platform requirements, for your deployment type.

      iii. Download the files for the operators required by IBM Cloud Pak foundational services (common services):

      ```
      oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
      ```

      iv. Generate the catalog sources:

      ```
      oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
      ```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources:

      ```
      cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
      ```

      This command might not return anything if there are no architecture-specific catalog sources.

   b. Apply the catalog sources:
      i. Apply the catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

ii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

9. Create the `apiconnect` subscription.
   a. Create the `apiconnect` subscription with the appropriate channel by creating a file called `apic-sub.yaml` and pasting in the following contents, updating the namespace as required:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: <namespace>
spec:
  channel: v5.1
  name: ibm-apiconnect
  source: ibm-apiconnect-catalog
  sourceNamespace: openshift-marketplace
```

   Where `<namespace>` is one of the following values:
   - `openshift-operators` if you are installing the operator in all namespaces
   - The namespace created in step 3 if you are installing the operator in a single namespace

   b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```

   c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
   If any operators such as `ibm-apiconnect` or `IBM DataPower Gateway` show the status of "Upgrade available", approve the upgrade by completing the following steps:

   i. Click Upgrade available.
   ii. Click Preview InstallPlan.
   iii. Click Approve.
   iv. Wait for the `IBM API Connect` and `IBM DataPower Gateway` operators to install.

   The `IBM DataPower Gateway` operator is a prerequisite to API Connect and must not be removed.

10. Install cert manager by completing the following steps:
    a. Log in to the OpenShift Container Platform web console.

    b. Navigate to Operators > OperatorHub.

    c. In the filter box, type: `cert-manager Operator for Red Hat OpenShift`.

    d. Select cert-manager Operator for Red Hat OpenShift and click Install.

    e. On the Install Operator page, complete the following steps:
       i. Update the Update channel if needed. The channel defaults to `stable-v1`, which installs the latest stable release of the cert-manager Operator for Red Hat OpenShift.

       ii. Select the Installed Namespace for the operator.
       The default operator namespace is `cert-manager-operator`; if that namespace doesn't exist, it is created for you.

       iii. Select an Update approval strategy:
          - Automatic: allow Operator Lifecycle Manager (OLM) to automatically update the operator when a new version is available.
          - Manual: require a user with the appropriate credentials to approve all operator updates.

       iv. Click Install.

    f. Verify the cert-manager installation by completing the following steps:
       i. Click to Operators > Installed Operators.

       ii. Verify that `cert-manager Operator for Red Hat OpenShift` is listed with a Status of `Succeeded` in the `cert-manager-operator` namespace.

       iii. Verify that cert-manager pods are up and running by running the following command:

```
oc get pods -n cert-manager
```

       For a successful installation, the response looks like the following example:

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-bd7fbb9fc-wvbbt               1/1     Running   0          3m39s
cert-manager-cainjector-56cc5f9868-7g9z7   1/1     Running   0          4m5s
cert-manager-webhook-d4f79d7f7-9dg9w       1/1     Running   0          4m9s
```

11. Create the `ibm-common-services-operator` subscription, if it does not already exist.
    a. Create a file called common-services-sub.yaml and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <namespace>
spec:
  channel: v4.3
  name: ibm-common-service-operator
  source: opencloud-operators
  sourceNamespace: openshift-marketplace
```

Where: ***<namespace>*** is one of the following values:

- **openshift-operators** if you are installing the operator in all namespaces
- The namespace created in step 3 if you are installing the operator in a single namespace

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as **ibm-apiconnect**, **IBM Cloud Pak foundational services**, or **Operand Deployment Lifecycle Manager** show the status of "Upgrade available", approve the upgrade by completing the following steps:

i. Click Upgrade available.
ii. Click Preview InstallPlan.
iii. Click Approve.
iv. Check the **ibm-common-services** namespace and ensure that all operators with a status of "Upgrade available" are approved.
v. Wait for the **IBM Cloud Pak foundational services** and **Operand Deployment Lifecycle Manager** operators to install.

Next topic: Setting up a certificate issuer

Setting up a certificate issuer

Create the cert-manager issuer for generating the certificates and secrets for use with API Connect.

Before you begin

To deploy API Connect, you must first create the cert-manager issuers and certificates that API Connect uses. On OpenShift cert-manager is bundled as a foundational service, so you do not need to install cert-manager itself. For more information on cert-manager: Key Concepts: Cert-manager, Issuers, and Secrets

Procedure

1. Create a file that is called **ingress-issuer-v1.yaml** and paste in the following contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  issuerRef:
    name: selfsigning-issuer
```

```
    kind: Issuer
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "ingress-ca"
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "portal-admin-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-client-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: analytics-ingestion-client
```

```
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "analytics-ingestion-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "gatewaycluster"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-service"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "gatewaycluster"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-peering"
```

2. Apply the file to your namespace with **`oc apply -f ingress-issuer-v1.yaml -n <apic_namespace>`**
3. Verify that the command installation succeeded by running the following command:

   `oc get certificates -n <apic_namespace>`

   All certificates created successfully:

```
NAME                          READY   SECRET                        AGE
analytics-ingestion-client    True    analytics-ingestion-client    70s
gateway-client-client         True    gateway-client-client         69s
gateway-peering               True    gateway-peering               69s
gateway-service               True    gateway-service               69s
ingress-ca                    True    ingress-ca                    71s
portal-admin-client           True    portal-admin-client           71s
```

Previous topic: Installing operators
Next topic: Installing the Management subsystem in a shared namespace

Installing the Management subsystem in a shared namespace

Install the Management subsystem by creating and applying the mgmt_cr.yaml file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. Preparing for installation
2. Installing operators
3. Setting up a certificate issuer

About this task

Use the OpenShift CLI to edit the custom resource template for the Management subsystem then apply the resource. Verify that the pods are up and running, and validate that you can connect to the API Connect Cloud Manager.

Procedure

1. Create a file that is called mgmt_cr.yaml and paste in the following content:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  name: management
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "management"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  portal:
    admin:
      secretName: portal-admin-client
  analytics:
    ingestion:
      secretName: analytics-ingestion-client
  gateway:
    client:
      secretName: gateway-client-client
  cloudManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: admin.$STACK_HOST
      secretName: cm-endpoint
  apiManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: manager.$STACK_HOST
      secretName: apim-endpoint
  platformAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: api.$STACK_HOST
      secretName: api-endpoint
  consumerAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: consumer.$STACK_HOST
      secretName: consumer-endpoint
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
```

```
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  databaseBackup:
    host: $S3_HOST
    path: $S3_PATH
    credentials: $S3_CREDENTIALS
    backupCerts: $S3_CA_CERT
    schedule: "$CRON_SCHEDULE"
    repoRetentionFull: $RETENTION_DAYS
    backups3URIStyle: $PATH_STYLE # host or virtual style is the default
```

2. Edit the YAML file and replace the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your Management subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```
cloudManagerEndpoint:
      < ... >
    hosts:
    - name: admin.myhost.subnet.example.com
      secret: cm-endpoint

  apiManagerEndpoint:
      < ... >
    hosts:
    - name: manager.myhost.subnet.example.com
      secret: apim-endpoint

  platformAPIEndpoint:
      < ... >
    hosts:
    - name: api.myhost.subnet.example.com
      secret: api-endpoint

  consumerAPIEndpoint:
      < ... >
    hosts:
    - name: consumer.myhost.subnet.example.com
      secret: consumer-endpoint
```

- Complete hostname customization
  Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

  For example, for **cloudManagerEndpoint**, you can replace **admin.$STACK_HOST** with **my.cloudmgr.myhost.subnet.example.com**, where **my.cloudmgr** replaces **admin**, and **myhost.subnet.example.com** replaces **STACK_HOST**. For example:

```
cloudManagerEndpoint:
      < ... >
    hosts:
    - name: my.cloudmgr.myhost.subnet.example.com
      secret: cm-endpoint
```

  You can do this for some or all of the host names, depending on your customization requirements.

$STORAGE_CLASS
The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**.

```
storageClassName: local-storage
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.

$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.

$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

4. Configure database backups for the Management subsystem.
   Set the properties in the **databaseBackup** backup section. See Configuring management database backups.
   Note: If you do not have an S3 object-store available, you can install the management subsystem with no backup settings configured, by adding this annotation to the CR:

```
metadata:
  annotations:
    apiconnect-operator/backups-not-configured: "true"
```

   If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
5. Install the Management subsystem by applying the modified CR with the following command:

```
oc apply -f mgmt_cr.yaml -n <management_namespace>
```

6. Verify that the Management subsystem is fully installed by running the following command:

```
oc get ManagementCluster -n <management_namespace>
```

   The installation is complete when the **READY** status is **True**, and the **SUMMARY** reports that all services are online:

```
NAME          READY   SUMMARY   VERSION    RECONCILED VERSION   AGE
management    True    16/16      <version>   <version-build>      7m17s
```

7. Verify that you can log in to the API Connect Cloud Manager UI:
   a. Determine the URL for the Cloud Manager UI by running the following command to view the API Connect endpoints:

```
oc get routes -n <management_namespace>
```

   b. Locate the **management-admin** endpoint, and note it down.
   c. Determine the Cloud Manager administrator password by running the following commands:

```
oc get secret -n <management-namespace> | grep management-admin-secret

oc get secret -n <management-namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -
d && echo ""
```

   d. Open a browser and go to the **management-admin-apic** endpoint.
   e. Log in to the Cloud Manager as **admin** with the administrator password.
8. Store all secrets and passwords in a safe place.
9. Backup your management subsystem database and infrastructure configuration: Backing up and restoring the management subsystem.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Setting up a certificate issuer
Next topic: Installing the Gateway subsystem in a shared namespace

Installing the Gateway subsystem in a shared namespace

Install the Gateway subsystem by creating and applying the apigateway_cr.yaml file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. Preparing for installation
2. Installing operators
3. Setting up a certificate issuer

About this task

Important: When you install the gateway subsystem, you must configure the gateway subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
Use the OpenShift CLI to edit the custom resource template for the Gateway, apply the resource, verify that the pods are up and running.

Procedure

1. Create the gateway admin secret with: **oc -n <gateway namespace> create secret generic admin-secret --from-literal=password=admin**
2. If you are installing the API Gateway (v6), create a yaml file that is called **apigateway_cr.yaml** and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv6
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv6"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: false
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: rgw.$STACK_HOST
      secretName: gwv6-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: rgwd.$STACK_HOST
      secretName: gwv6-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  tokenManagementService:
    enabled: true
    storage:
      storageClassName: $STORAGE_CLASS
      volumeSize: 30Gi
  adminUser:
    secretName: admin-secret
  # syslogConfig:
  #   enabled: false # if true, provide below details
  #   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
  #   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
  #   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

If you are installing the v5 Gateway, create a yaml file that is called v5gateway_cr.yaml and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv5
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv5"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: true
```

```
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: gw.$STACK_HOST
      secretName: gwv5-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: gwd.$STACK_HOST
      secretName: gwv5-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  adminUser:
    secretName: admin-secret
  # syslogConfig:
  #   enabled: false # if true, provide below details
  #   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
  #   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
  #   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

3. Edit the YAML file and set the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your Gateway subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

  ```
  gatewayEndpoint:
        < ... >
      hosts:
      - name: rgw.myhost.subnet.example.com
        secret: gwv6-endpoint

    gatewayManagerEndpoint:
        < ... >
      hosts:
      - name: rgwd.myhost.subnet.example.com
        secret: gwv6-manager-endpoint
  ```

- Complete hostname customization
  Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

  For example, for **gatewayEndpoint**, you can replace **rgw.$STACK_HOST** with **my.gateway.endpoint.myhost.subnet.example.com**, where **my.gateway.endpoint** replaces **rgw**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

  ```
  gatewayEndpoint:
        < ... >
      hosts:
      - name: my.gateway.endpoint.myhost.subnet.example.com
        secret: gwv6-endpoint
  ```

  You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS
The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**.

```
storageClassName: local-storage
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.
$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.
$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

syslogConfig
To enable this feature uncomment this section, set **enabled: true**, and set the properties:

$DATAPOWER_SYSLOG_TCP_REMOTE_HOST
`remoteHost: <remote.host>`
String. The hostname of the server for `syslogConfig`.

$DATAPOWER_SYSLOG_TCP_REMOTE_PORT
`remotePort: <remote_port_number>`
Integer. The port number of the server for `syslogConfig`.

$DATAPOWER_SYSLOG_TCP_TLS_SECRET
`secretName: <secret>`
String. The TLS secret name of the server for `syslogConfig`.

4. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the `YAML` tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

5. Optional: If applicable to your deployment, update the gateway CR file to configure `DataPowerService` APIs to customize your DataPower deployment. See Customizing a DataPower deployment.
6. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add `mtlsValidateClient` to the `spec` section:

```
spec:
  mtlsValidateClient: true
```

This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.
Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.
7. Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwksUrl`.

```
spec:
  ...
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the `jwksUrl`, describe the management CR and check the `status:` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.
8. Optional: Update the gateway CR file to enable autoscaling of gateway pods to ensure high availability of DataPower pods. See Enabling autoscaling of gateway pods.
9. Install the Gateway Custom Resource by applying the yaml file:
For a DataPower® API Gateway, run the following command:

```
oc apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
oc apply -f v5cgateway_cr.yaml -n <namespace>
```

10. To verify that the Gateway subsystem is fully installed, run the following command:

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

The installation is complete when the `READY` status is `True`, and the `SUMMARY` reports that all services are online (`2/2`) for all the Gateway subsystems that were installed. Example:

```
NAME   READY   SUMMARY   VERSION     RECONCILED VERSION   AGE
gwv6   True    2/2       <version>   <version-build>      7m32s
```

There is no need to wait for the `READY` status to be `True` before moving on to next Subsystem installation.
11. Verify that the gateway is ready to be registered with the Management subsystem by calling the `health` api on the `gatewayManagerEndpoint.hosts.name` defined in your yaml file: `https://gwd.$STACK_HOST/health`.

```
curl -k https://gwd.example.com/health
{"status":"ok"}
```

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing the Management subsystem in a shared namespace
Next topic: Installing the Portal subsystem in a shared namespace

Installing the Portal subsystem in a shared namespace

Install the Portal subsystem by creating and applying the portal_cr.yaml file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. Preparing for installation
2. Installing operators
3. Setting up a certificate issuer

About this task

Important: When you install the portal subsystem, you must configure the portal subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
Use the OpenShift CLI to edit the custom resource template for the Portal subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called **portal_cr.yaml** and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalCluster
metadata:
  name: portal
  labels: {
    app.kubernetes.io/instance: "portal",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtConsumerEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca
  portalAdminEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: api.portal.$STACK_HOST
      secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: portal.$STACK_HOST
      secretName: portal-web
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
  databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 12Gi
  webVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
  backupVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 120Gi
  adminVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 20Gi
  certVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 4Gi
```

```
  adminClientSubjectDN: CN=portal-admin-client,O=cert-manager
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
```

2. Edit the yaml file and set the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your portal subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **$STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of **myhost.subnet.example.com**:

  ```
  portalAdminEndpoint:
      < ... >
      hosts:
      - name: api.portal.myhost.subnet.example.com
        secret: portal-admin

    portalUIEndpoint:
      < ... >
      hosts:
      - name: portal.myhost.subnet.example.com
        secret: portal-web
  ```

- Complete hostname customization
  Change both the predefined prefixes and the **$STACK_HOST** subdomain to match your desired hostnames.

  For example, for **portalAdminEndpoint**, you can replace **api.portal.$STACK_HOST** with **my.api.portal.myhost.subnet.example.com**, where **my.api.portal** replaces **api.portal**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

  ```
  portalAdminEndpoint:
        < ... >
      hosts:
      - name: my.api.portal.myhost.subnet.example.com
        secret: cm-endpoint
  ```

  You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS
The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**. For example, for local storage:

```
databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 12Gi
webVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
backupVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 300Gi
adminVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 20Gi
certVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 4Gi
```

Note:

- The **databaseVolumeClaimTemplate** has a minimum value of 30Gi. The **webVolumeClaimTemplate** has a minimum value of 20Gi. The **backupVolumeClaimTemplate** has a minimum value of 30Gi. However, all three need to be sized according to the installation. For more information on planning your portal PV requirements, see Portal subsystem.
- The **adminVolumeClaimTemplate** and **databaseLogsVolumeClaimTemplate** in this example have their minimum values and you do not need to change them.
- The **certVolumeClaimTemplate** is used by the nginx pod and in 2DCDR mode it is also by the **tunnel**, **db-remote** and **www-remote** pods. It can be sized small, just like the **adminVolumeClaimTemplate**, and does not need to be increased based on the number of sites.

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.

$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.

$LICENSE_ID
: Set `license:` to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

4. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties `mtlsValidateClient` and `jwksUrl`.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the `jwksUrl`, describe the management CR and check the `status:` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.

5. Install the Portal subsystem by applying the modified CR with the following command:

```
oc apply -f portal_cr.yaml -n <namespace>
```

6. Verify that the Portal subsystem is fully installed:

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

The installation is complete when `STATUS` reports `Running`, `Ready` reports that all services are online (`3/3`), and `MESSAGE` reports "Ready for API Cloud Manager service registration", for example:

```
NAME            READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE                                  AGE
portal          3/3     Running   <version>  <version>            Ready for API Cloud Manager service registration   2h
```

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

7. Verify that you can access the portal web endpoint by browsing to the `portalUIEndpoint.hosts.name` as configured in your yaml file:

```
curl -k https://portal.$STACK_HOST
<!DOCTYPE html>
<html>
<head>
<title>Developer Portal is ready to create sites!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>Developer Portal is ready to create sites!</h1>
<p>If you see this page, the web server is successfully installed and
working. Further configuration is required.</p>
</body>
</html>
```

8. It is highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.
The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource after the installation is complete:

   a. Obtain the name of the encryptionSecret:

   ```
   oc get ptl portal -o yaml -n <namespace> | grep encryptionSecret
   encryptionSecret: portal-enc-key
   ```

   The name of encryption secret is `portal-enc-key`.

   b. Save the secret yaml in a safe location:

   ```
   oc get secret portal-enc-key -o yaml -n <namespace> > portal-enc-key.yaml
   ```

9. Backup the Portal installation. See Backing up and restoring the developer portal.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing the Gateway subsystem in a shared namespace
Next topic: Installing the Analytics subsystem in a shared namespace

Installing the Analytics subsystem in a shared namespace

Install the analytics subsystem by creating and applying the analytics_cr.yaml file.

Before you begin

Complete the following tasks to prepare for deploying API Connect:

1. Preparing for installation
2. Installing operators
3. Setting up a certificate issuer

About this task

Use the OpenShift CLI to edit the custom resource template for the analytics subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called **analytics_cr.yaml** and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
metadata:
  name: analytics
  labels: {
    app.kubernetes.io/instance: "analytics",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics"
  }
spec:
  version: $APP_PRODUCT_VERSION
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  profile: $PROFILE
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  ingestion:
    endpoint:
      annotations:
        cert-manager.io/issuer: ingress-issuer
      hosts:
      - name: ai.$STACK_HOST
        secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
  storage:
    type: shared # change to 'dedicated' if you want separate master and storage pods. Three-replica deployments only.
    shared:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
        volumeSize: $DATA_VOLUME_SIZE
    # master: # uncomment this section if you set storage.type = dedicated.
    #   volumeClaimTemplate:
    #     storageClassName: $STORAGE_CLASS
```

2. Edit the YAML file and set the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

`version: <version_number>`

Example version number: 10.0.7.0

$PROFILE
Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete hostname:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **$STACK_HOST** with the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

  ```
  ingestion:
    endpoint:
       < ... >
      hosts:
      - name: ai.myhost.subnet.example.com
        secret: analytics-ai-endpoint
       < .... >
  ```

- Complete hostname customization
  Change both the predefined prefixes and the **$STACK_HOST** subdomain to match your desired hostnames.

  For example, you can replace **$STACK_HOST** with **my.analytics.ingestion.myhost.subnet.example.com**, where **my.analytics.ingestion** replaces **ai**, and **myhost.subnet.example.com** replaces **STACK_HOST**.

  ```
  ingestion:
     endpoint:
       < ... >
       hosts:
       - name: my.analytics.ingestion.myhost.subnet.example.com
         secret: analytics-ai-endpoint
        < ... >
  ```

$STORAGE_CLASS
The Kubernetes storage class to be used for persistent volume claims. For more information, see Analytics preinstallation planning. Find the available storage classes in the target cluster by running the following command: **oc get sc**. Example:

```
storage:
    type: shared
    shared:
      volumeClaimTemplate:
        storageClassName: ceph-block
```

$DATA_VOLUME_SIZE
Size of storage allocated for data. To estimate the storage space you require, see Estimating storage requirements. If you are unable to estimate your storage requirement then set it to 500Gi.

```
storage:
    type: shared
    shared:
      volumeClaimTemplate:
        storageClassName: ceph-block
        volumeSize: 500Gi
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.
$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.
$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

   ```
   metadata:
     annotations:
       apiconnect-operator/cp4i: "false"
   ```

   This annotation disables all integration with Cloud Pak for Integration.

4. Optional: Three replica deployments only: Enable dedicated storage.
   If you want to use dedicated storage, update the **spec.storage** section of the yaml file as follows:

   ```
   storage:
     type: dedicated
     shared:
       volumeClaimTemplate:
         storageClassName: $STORAGE_CLASS
         volumeSize: $DATA_VOLUME_SIZE
     master:
       volumeClaimTemplate:
         storageClassName: $STORAGE_CLASS
   ```

   For more information about dedicated storage, see dedicated or shared storage.
5. Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties **mtlsValidateClient** and **jwksUrl**.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the **jwksUrl**, describe the management CR and check the **status:** section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.
Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.

6. Install the analytics subsystem by applying the modified CR with the following command:

```
oc apply -f analytics_cr.yaml -n <namespace>
```

7. Verify that the analytics subsystem is fully installed:

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

The installation is complete when **READY** shows all pods running (**n/n**), and the **STATUS** reports **Running**. Example:

```
NAME         READY   STATUS    VERSION          RECONCILED VERSION   AGE
analytics    n/n     Running   10.0.7.0   10.0.7.0-1281    86m
```

It is not necessary to wait for analytics installation to complete before you move on to the next subsystem installation.

8. Backup your analytics subsystem and configure scheduled analytics database backups: Backing up and restoring the analytics subsystem.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing the Portal subsystem in a shared namespace

Installing API Connect subsystems in different namespaces or environments on OpenShift

Install API Connect subsystems in different namespaces, for example Management subsystem in a namespace that is called **apicmgmt**, Gateway in a namespace called **apicgway**. These same steps can also be used for installing subsystems in different environments.

About this task

Install API Connect subsystems where you have separate namespaces for each subsystem.

Air-gapped installations: If you are installing API Connect in an offline environment, first follow the steps for configuring the catalog sources as described here: Air-gapped installation.

Attention: If you want to configure FIPS support for API Connect, you must enable FIPS on the cluster as part of the initial OpenShift deployment. You cannot modify an existing cluster to enable FIPS support. For more information, see Configuring FIPS support on OpenShift.

1. Installing operators
   Install operators for API Connect, DataPower, and IBM Cloud Pak foundational services so that you can deploy API Connect subsystems across multiple namespaces on Red Hat OpenShift. If you deploy API Connect subsystems in different environments, then the corresponding operator must be installed in each environment; for example, install the DataPower operator in your Gateway environment, and the API Connect operator in your Portal environment.
2. Installing the Management subsystem
   Install the Management subsystem by creating and applying the ingress-issuer and mgmt_cr yaml files.
3. Extracting the Management ingress-ca certificates
   In order for the Management subsystem to be able to communicate with subsystems in different namespaces, these subsystems need to have the same ingress-ca certificates as the Management subsystem.
4. Installing the Gateway subsystem
   Install the Gateway subsystem by creating and applying the ingress-ca, common-issuer, gateway-certs and gateway_cr yaml files.
5. Installing the Portal subsystem
   Install the Portal subsystem by creating and applying the ingress-ca, common-issuer, and portal_cr yaml files.
6. Installing the Analytics subsystem
   Install the Analytics subsystem by creating and applying the ingress-ca, common-issuer, and analytics_cr yaml files.

Installing operators

Install operators for API Connect, DataPower, and IBM Cloud Pak foundational services so that you can deploy API Connect subsystems across multiple namespaces on Red Hat OpenShift. If you deploy API Connect subsystems in different environments, then the corresponding operator must be installed in each environment; for example, install the DataPower operator in your Gateway environment, and the API Connect operator in your Portal environment.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.7.0	v5.1	5.1.0

Note: There can be only one instance of the API Connect operator in your OpenShift environment. You cannot have multiple namespaces each with their own API Connect operator.
You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

Procedure

1. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step #inst_ocp_cp4i_ops_subs__d322e179.
   The **ibm-pak** plug-in enables you to access hosted product images, and to run **oc ibm-pak** commands against the cluster. To confirm that **ibm-pak** is installed, run the following command and verify that the response lists the command usage:

   ```
   oc ibm-pak --help
   ```

2. Obtain an entitlement key for the Entitled Registry:
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step #inst_ocp_cp4i_ops_subs__d322e179.
   a. Log in to the IBM Container Library.
   b. In the Container software library, select Get entitlement key.
   c. After the Access your container software heading, click Copy key.
   d. Copy the key to a safe location.
3. Create a pull secret in the namespaces where you want to install API Connect.
   a. Open the Red Hat OpenShift web console, click Workloads > Secrets.
   b. Ensure that the Project is set to the namespace where you intend to install API Connect.
   c. Click Create and select Image pull secret.
   d. Set the following parameters for the secret:
      - Set Secret name to **ibm-entitlement-key**.
      - Set Authentication type to **Image registry credentials**.
      - Set Registry server address to **cp.icr.io**.
      - Set Username to **cp**.
      - Set Password to the entitlement key generated in Step 2.
      - Click Create to create the secret.

   Note: If your Red Hat OpenShift platform is Red Hat OpenShift on IBM Cloud (ROKS), you might need to reload your worker nodes after the image pull secret is created. You can reload the worker nodes either from the Red Hat OpenShift web console or from the command line with: **ibmcloud oc worker reload**. For information on using the **reload** command, see the Red Hat OpenShift on IBM Cloud documentation.
4. Add the API Connect catalog sources to your cluster.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   Adding catalog sources to your Red Hat OpenShift cluster adds the IBM operators to the list of operators you can install. For most connected and air-gapped clusters, applying individual catalog sources is the most effective way to fully control software versioning on the cluster.

   a. Generate the catalog source files:
      i. Log into your cluster using the **oc login** command and your user credentials:

      ```
      oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
      ```

      ii. Export the variables for the command line to use:

      ```
      export CASE_NAME=ibm-apiconnect
      export CASE_VERSION=5.1.0
      export ARCH=amd64
      ```

      For information on API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

      iii. Download the files for the operators required by API Connect:

      ```
      oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
      ```

      iv. Generate the catalog sources for API Connect:

      ```
      oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
      ```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources (run both commands):

      ```
      cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

      cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
      ```

   b. Apply the catalog sources:
      i. Apply the catalog sources (run both commands):

      ```
      oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
      ```

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

ii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

5. Add the IBM Cloud Pak foundational services (common services) catalog sources to your cluster.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   a. Generate the catalog source files:
      i. Log into your cluster using the `oc login` command and your user credentials:

```
oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
```

      ii. Export the variables for the command line to use; substituting the appropriate CASE version:

```
export CASE_NAME=ibm-cp-common-services
export CASE_VERSION=<case version>
export ARCH=amd64
```

      Set *<case version>* to the value specified for *Foundational services channel version* in Operator, CASE, and platform requirements, for your deployment type.

      iii. Download the files for the operators required by IBM Cloud Pak foundational services (common services):

```
oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
```

      iv. Generate the catalog sources:

```
oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

      This command might not return anything if there are no architecture-specific catalog sources.

   b. Apply the catalog sources:
      i. Apply the catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

      ii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

6. Create the `apiconnect` subscription.
   a. Create the `apiconnect` subscription with the appropriate channel by creating a file called `apic-sub.yaml` and pasting in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: openshift-operators
spec:
  channel: v5.1
  name: ibm-apiconnect
  source: ibm-apiconnect-catalog
  sourceNamespace: openshift-marketplace
```

   b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```

   c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
   If any operators such as `ibm-apiconnect` or `IBM DataPower Gateway` show the status of "Upgrade available", approve the upgrade by completing the following steps:

      i. Click Upgrade available.
      ii. Click Preview InstallPlan.
      iii. Click Approve.
      iv. Wait for the `IBM API Connect` and `IBM DataPower Gateway` operators to install.

   The `IBM DataPower Gateway` operator is a prerequisite to API Connect and must not be removed.

7. Install cert manager by completing the following steps:
   a. Log in to the OpenShift Container Platform web console.

   b. Navigate to Operators > OperatorHub.

   c. In the filter box, type: `cert-manager Operator for Red Hat OpenShift`.

   d. Select cert-manager Operator for Red Hat OpenShift and click Install.

   e. On the Install Operator page, complete the following steps:
      i. Update the Update channel if needed. The channel defaults to `stable-v1`, which installs the latest stable release of the cert-manager Operator for Red Hat OpenShift.

      ii. Select the Installed Namespace for the operator.
      The default operator namespace is `cert-manager-operator`; if that namespace doesn't exist, it is created for you.

iii. Select an Update approval strategy:
- Automatic: allow Operator Lifecycle Manager (OLM) to automatically update the operator when a new version is available.
- Manual: require a user with the appropriate credentials to approve all operator updates.

iv. Click Install.

f. Verify the cert-manager installation by completing the following steps:

i. Click to Operators > Installed Operators.

ii. Verify that `cert-manager Operator for Red Hat OpenShift` is listed with a Status of `Succeeded` in the `cert-manager-operator` namespace.

iii. Verify that cert-manager pods are up and running by running the following command:

```
oc get pods -n cert-manager
```

For a successful installation, the response looks like the following example:

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-bd7fbb9fc-wvbbt               1/1     Running   0          3m39s
cert-manager-cainjector-56cc5f9868-7g9z7   1/1     Running   0          4m5s
cert-manager-webhook-d4f79d7f7-9dg9w       1/1     Running   0          4m9s
```

8. Create the `ibm-common-services-operator` subscription, if it does not already exist.

a. Create a file called common-services-sub.yaml and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: openshift-operators
spec:
  channel: v4.3
  name: ibm-common-service-operator
  source: opencloud-operators
  sourceNamespace: openshift-marketplace
```

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as `ibm-apiconnect`, `IBM Cloud Pak foundational services`, or `Operand Deployment Lifecycle Manager` show the status of "Upgrade available", approve the upgrade by completing the following steps:

i. Click Upgrade available.
ii. Click Preview InstallPlan.
iii. Click Approve.
iv. Check the `ibm-common-services` namespace and ensure that all operators with a status of "Upgrade available" are approved.
v. Wait for the `IBM Cloud Pak foundational services` and `Operand Deployment Lifecycle Manager` operators to install.

9. Create the namespaces for your API Connect subsystems either by selecting Project > Create Project in the UI, or with the command:

```
oc create ns <APIC-namespace>
```

Note: Red Hat OpenShift restricts the use of default namespaces for installing non-cluster services. The following namespaces cannot be used to install API Connect:

- `default`
- `kube-system`
- `kube-public`
- `openshift-node`
- `openshift-infra`
- `openshift`

What to do next

Proceed to install the API Connect subsystems as explained in: Installing the Management subsystem

Next topic: Installing the Management subsystem

Installing the Management subsystem

Install the Management subsystem by creating and applying the ingress-issuer and mgmt_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Create a file called `ingress-issuer-management.yaml` and paste in the following:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
```

```
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
```

```
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

2. Apply the YAML file with: `oc apply -f ingress-issuer-management.yaml -n <management namespace>`
3. Confirm that the issuers were created and are in ready state:

```
oc get issuers -n <management namespace>
NAME                  READY
ingress-issuer        True
selfsigning-issuer    True
```

Procedure

1. Create a file that is called mgmt_cr.yaml and paste in the following content:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementCluster
metadata:
  name: management
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "management"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  portal:
    admin:
      secretName: portal-admin-client
  analytics:
    ingestion:
      secretName: analytics-ingestion-client
  gateway:
    client:
      secretName: gateway-client-client
```

```
  cloudManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: admin.$STACK_HOST
      secretName: cm-endpoint
  apiManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: manager.$STACK_HOST
      secretName: apim-endpoint
  platformAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: api.$STACK_HOST
      secretName: api-endpoint
  consumerAPIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: consumer.$STACK_HOST
      secretName: consumer-endpoint
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  databaseBackup:
    host: $S3_HOST
    path: $S3_PATH
    credentials: $S3_CREDENTIALS
    backupCerts: $S3_CA_CERT
    schedule: "$CRON_SCHEDULE"
    repoRetentionFull: $RETENTION_DAYS
    backups3URIStyle: $PATH_STYLE # host or virtual style is the default
```

2. Edit the YAML file and replace the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your Management subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```
cloudManagerEndpoint:
      < ... >
    hosts:
    - name: admin.myhost.subnet.example.com
      secret: cm-endpoint

  apiManagerEndpoint:
      < ... >
    hosts:
    - name: manager.myhost.subnet.example.com
      secret: apim-endpoint

  platformAPIEndpoint:
      < ... >
    hosts:
    - name: api.myhost.subnet.example.com
      secret: api-endpoint

  consumerAPIEndpoint:
      < ... >
    hosts:
    - name: consumer.myhost.subnet.example.com
      secret: consumer-endpoint
```

- Complete hostname customization
  Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

  For example, for **cloudManagerEndpoint**, you can replace **admin.$STACK_HOST** with **my.cloudmgr.myhost.subnet.example.com**, where **my.cloudmgr** replaces **admin**, and **myhost.subnet.example.com** replaces **STACK_HOST**. For example:

```
cloudManagerEndpoint:
      < ... >
   hosts:
   - name: my.cloudmgr.myhost.subnet.example.com
     secret: cm-endpoint
```

You can do this for some or all of the host names, depending on your customization requirements.

$STORAGE_CLASS
The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**.

```
storageClassName: local-storage
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.
$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.
$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

4. Configure database backups for the Management subsystem.
Set the properties in the **databaseBackup** backup section. See Configuring management database backups.
Note: If you do not have an S3 object-store available, you can install the management subsystem with no backup settings configured, by adding this annotation to the CR:

```
metadata:
  annotations:
    apiconnect-operator/backups-not-configured: "true"
```

If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
5. Install the Management subsystem by applying the modified CR with the following command:

```
oc apply -f mgmt_cr.yaml -n <management_namespace>
```

6. Verify that the Management subsystem is fully installed by running the following command:

```
oc get ManagementCluster -n <management_namespace>
```

The installation is complete when the **READY** status is **True**, and the **SUMMARY** reports that all services are online:

```
NAME          READY   SUMMARY   VERSION    RECONCILED VERSION   AGE
management    True    16/16       <version>   <version-build>       7m17s
```

7. Verify that you can log in to the API Connect Cloud Manager UI:
   a. Determine the URL for the Cloud Manager UI by running the following command to view the API Connect endpoints:

   ```
   oc get routes -n <management_namespace>
   ```

   b. Locate the **management-admin** endpoint, and note it down.
   c. Determine the Cloud Manager administrator password by running the following commands:

   ```
   oc get secret -n <management-namespace> | grep management-admin-secret

   oc get secret -n <management-namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -
   d && echo ""
   ```

   d. Open a browser and go to the **management-admin-apic** endpoint.
   e. Log in to the Cloud Manager as **admin** with the administrator password.
8. Store all secrets and passwords in a safe place.
9. Backup your management subsystem database and infrastructure configuration: Backing up and restoring the management subsystem.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing operators
Next topic: Extracting the Management ingress-ca certificates

Extracting the Management ingress-ca certificates

In order for the Management subsystem to be able to communicate with subsystems in different namespaces, these subsystems need to have the same ingress-ca certificates as the Management subsystem.

About this task

This task describes how to extract the ingress-ca certificates to a yaml file, ready for applying to the subsystems that are in different namespaces.

Procedure

1. Export the ingress-ca secret to a yaml file with: `oc -n <management namespace> get secret ingress-ca -o yaml > ingress-ca.yaml`
2. Edit the ingress-ca.yaml file and remove the following properties: metadata.creationTimestamp, metadata.namespace,metadata.resourceVersion, metadata.uid, metadata.selfLink
3. Keep this file safe. You will need to apply it in the namespaces of your other subsystems so that they can communicate with the Management subsystem.

Previous topic: Installing the Management subsystem
Next topic: Installing the Gateway subsystem

Installing the Gateway subsystem

Install the Gateway subsystem by creating and applying the ingress-ca, common-issuer, gateway-certs and gateway_cr yaml files.

Before you begin

Important: When you install the gateway subsystem, you must configure the gateway subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Using the file obtained from Extracting the Management ingress-ca certificates, run: `oc apply -f ingress-ca.yaml -n <gateway namespace>`
2. Create a file called `common-issuer-and-gateway-certs.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
```

```
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
```

3. Apply the above yaml file with: oc apply -f common-issuer-and-gateway-certs.yaml -n <gateway namespace>
4. Confirm that the issuers were created and are in ready state:

```
oc get issuers -n <gateway namespace>
NAME                  READY
ingress-issuer        True
selfsigning-issuer    True
```

5. Confirm that the gateway secrets were created with `oc get secrets -n <gateway namespace>`:

```
oc get secrets -n <gateway namespace>
NAME                          TYPE                                  DATA   AGE
...
gateway-peering               kubernetes.io/tls                     3      2m
gateway-service               kubernetes.io/tls                     3      2m3s
...
```

About this task

Use the OpenShift CLI to edit the custom resource template for the Gateway, apply the resource, verify that the pods are up and running.

Procedure

1. Create the gateway admin secret with: `oc -n <gateway namespace> create secret generic admin-secret --from-literal=password=admin`
2. If you are installing the API Gateway (v6), create a yaml file that is called `apigateway_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv6
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv6"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: false
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
```

```
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: rgw.$STACK_HOST
      secretName: gwv6-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: rgwd.$STACK_HOST
      secretName: gwv6-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  tokenManagementService:
    enabled: true
    storage:
      storageClassName: $STORAGE_CLASS
      volumeSize: 30Gi
  adminUser:
    secretName: admin-secret
  # syslogConfig:
  #   enabled: false # if true, provide below details
  #   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
  #   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
  #   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

If you are installing the v5 Gateway, create a yaml file that is called v5gateway_cr.yaml and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: gateway.apiconnect.ibm.com/v1beta1
kind: GatewayCluster
metadata:
  name: gwv5
  labels: {
    app.kubernetes.io/instance: "gateway",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gwv5"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  apicGatewayServiceV5CompatibilityMode: true
  gatewayEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: gw.$STACK_HOST
      secretName: gwv5-endpoint
  gatewayManagerEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: gwd.$STACK_HOST
      secretName: gwv5-manager-endpoint
  apicGatewayServiceTLS:
    secretName: gateway-service
  apicGatewayPeeringTLS:
    secretName: gateway-peering
  datapowerLogLevel: 3
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  adminUser:
    secretName: admin-secret
  # syslogConfig:
  #   enabled: false # if true, provide below details
  #   remoteHost: $DATAPOWER_SYSLOG_TCP_REMOTE_HOST # must be a string
  #   remotePort: $DATAPOWER_SYSLOG_TCP_REMOTE_PORT # must be an int
  #   secretName: $DATAPOWER_SYSLOG_TCP_TLS_SECRET # must be a string
```

3. Edit the YAML file and set the variables:

```
$APP_PRODUCT_VERSION
```

API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your Gateway subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

  ```
  gatewayEndpoint:
        < ... >
    hosts:
    - name: rgw.myhost.subnet.example.com
      secret: gwv6-endpoint

  gatewayManagerEndpoint:
        < ... >
    hosts:
    - name: rgwd.myhost.subnet.example.com
      secret: gwv6-manager-endpoint
  ```

- Complete hostname customization
  Change both the predefined prefixes and the **STACK_HOST** subdomain to match your desired hostnames.

  For example, for **gatewayEndpoint**, you can replace **rgw.$STACK_HOST** with **my.gateway.endpoint.myhost.subnet.example.com**, where **my.gateway.endpoint** replaces **rgw**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

  ```
  gatewayEndpoint:
        < ... >
    hosts:
    - name: my.gateway.endpoint.myhost.subnet.example.com
      secret: gwv6-endpoint
  ```

  You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS
The OCP storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**.

```
storageClassName: local-storage
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.
$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.
$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

syslogConfig
To enable this feature uncomment this section, set **enabled: true**, and set the properties:
$DATAPOWER_SYSLOG_TCP_REMOTE_HOST
remoteHost: <remote.host>
String. The hostname of the server for **syslogConfig**.

$DATAPOWER_SYSLOG_TCP_REMOTE_PORT
remotePort: <remote_port_number>
Integer. The port number of the server for **syslogConfig**.

$DATAPOWER_SYSLOG_TCP_TLS_SECRET
secretName: <secret>
String. The TLS secret name of the server for **syslogConfig**.

4. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

   ```
   metadata:
     annotations:
       apiconnect-operator/cp4i: "false"
   ```

   This annotation disables all integration with Cloud Pak for Integration.

5. Optional: If applicable to your deployment, update the gateway CR file to configure **DataPowerService** APIs to customize your DataPower deployment. See Customizing a DataPower deployment.
6. To enable mutual TLS between the Management client and the Gateway service's manager endpoint, add **mtlsValidateClient** to the **spec** section:

   ```
   spec:
     mtlsValidateClient: true
   ```

   This ensures that the gateway service authenticates incoming requests from the API Manager, such as gateway service registration, and publishing APIs to the gateway service. Specifically, the gateway service requires that incoming requests present a certificate that was signed by the same CA that was used to sign the

gateway service management endpoint. The gateway service management endpoint secret is specified under `gatewayManagerEndpoint.hosts.secretName`. The API Manager's gateway client's TLS credentials are specified in the ManagementCluster CR under `gateway.client.secretName`.
Note: In releases previous to 10.0.5.3, this property is called `validateApimClient`.

7. Optional: If you want to enable JWT security for communications between the management and gateway, add and set the property `jwksUrl`.

```
spec:
  ...
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the `jwksUrl`, describe the management CR and check the `status:` section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.

8. Optional: Update the gateway CR file to enable autoscaling of gateway pods to ensure high availability of DataPower pods. See Enabling autoscaling of gateway pods.
9. Install the Gateway Custom Resource by applying the yaml file:
For a DataPower® API Gateway, run the following command:

```
oc apply -f apigateway_cr.yaml -n <namespace>
```

For a DataPower Gateway (v5 compatible), run the following command:

```
oc apply -f v5cgateway_cr.yaml -n <namespace>
```

10. To verify that the Gateway subsystem is fully installed, run the following command:

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

The installation is complete when the `READY` status is `True`, and the `SUMMARY` reports that all services are online (`2/2`) for all the Gateway subsystems that were installed. Example:

```
NAME   READY   SUMMARY   VERSION     RECONCILED VERSION    AGE
gwv6   True    2/2       <version>   <version-build>       7m32s
```

There is no need to wait for the `READY` status to be `True` before moving on to next Subsystem installation.

11. Verify that the gateway is ready to be registered with the Management subsystem by calling the `health` api on the `gatewayManagerEndpoint.hosts.name` defined in your yaml file: `https://gwd.$STACK_HOST/health`.

```
curl -k https://gwd.example.com/health
{"status":"ok"}
```

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Extracting the Management ingress-ca certificates
Next topic: Installing the Portal subsystem

Installing the Portal subsystem

Install the Portal subsystem by creating and applying the ingress-ca, common-issuer, and portal_cr yaml files.

Before you begin

Important: When you install the portal subsystem, you must configure the portal subsystem so that communication from the management subsystem is secured with either mTLS or JWT. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.
When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Using the file obtained from Extracting the Management ingress-ca certificates, run: `oc apply -f ingress-ca.yaml -n <portal namespace>`
2. Create a file called `common-issuer.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
```

```
---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
```

3. Apply the above yaml file with: `oc apply -f common-issuer.yaml -n <portal namespace>`
4. Confirm that the issuers were created and are in ready state:

```
oc get issuers -n <portal namespace>
NAME                READY
ingress-issuer      True
selfsigning-issuer  True
```

About this task

Use the OpenShift CLI to edit the custom resource template for the Portal subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called `portal_cr.yaml` and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#


apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalCluster
metadata:
  name: portal
  labels: {
    app.kubernetes.io/instance: "portal",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal"
  }
spec:
  version: $APP_PRODUCT_VERSION
  profile: $PROFILE
  mgmtPlatformEndpointCASecret:
    secretName: ingress-ca
  mgmtConsumerEndpointCASecret:
    secretName: ingress-ca
  mgmtPlatformEndpointSvcCASecret:
    secretName: management-ca
  mgmtConsumerEndpointSvcCASecret:
    secretName: management-ca
  portalAdminEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: api.portal.$STACK_HOST
      secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: ingress-issuer
    hosts:
    - name: portal.$STACK_HOST
```

```
    secretName: portal-web
  databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
  databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 12Gi
  webVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
  backupVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 120Gi
  adminVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 20Gi
  certVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 4Gi
  adminClientSubjectDN: CN=portal-admin-client,O=cert-manager
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
```

2. Edit the yaml file and set the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

```
version: <version_number>
```

Example version number: 10.0.7.0

$PROFILE
Specify your portal subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when you specify endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can do one of the following:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **$STACK_HOST** to be the desired ingress subdomain for the API Connect stack. For example, for a subdomain of **myhost.subnet.example.com**:

  ```
  portalAdminEndpoint:
      < ... >
      hosts:
      - name: api.portal.myhost.subnet.example.com
        secret: portal-admin

    portalUIEndpoint:
      < ... >
      hosts:
      - name: portal.myhost.subnet.example.com
        secret: portal-web
  ```

- Complete hostname customization
  Change both the predefined prefixes and the **$STACK_HOST** subdomain to match your desired hostnames.

  For example, for **portalAdminEndpoint**, you can replace **api.portal.$STACK_HOST** with **my.api.portal.myhost.subnet.example.com**, where **my.api.portal** replaces **api.portal**, and **myhost.subnet.example.com** replaces **$STACK_HOST**:

  ```
  portalAdminEndpoint:
        < ... >
      hosts:
      - name: my.api.portal.myhost.subnet.example.com
        secret: cm-endpoint
  ```

  You can do this for some or all of the hostnames, depending on your customization requirements.

$STORAGE_CLASS
The Kubernetes storage class to be used for Persistent Volume Claims. Find the available storage classes in the target cluster by running the following command: **oc get sc**. For example, for local storage:

```
databaseVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
databaseLogsVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 12Gi
webVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 200Gi
backupVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 300Gi
adminVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
```

```
    volumeSize: 20Gi
certVolumeClaimTemplate:
    storageClassName: $STORAGE_CLASS
    volumeSize: 4Gi
```

Note:

- The **databaseVolumeClaimTemplate** has a minimum value of 30Gi. The **webVolumeClaimTemplate** has a minimum value of 20Gi. The **backupVolumeClaimTemplate** has a minimum value of 30Gi. However, all three need to be sized according to the installation. For more information on planning your portal PV requirements, see Portal subsystem.
- The **adminVolumeClaimTemplate** and **databaseLogsVolumeClaimTemplate** in this example have their minimum values and you do not need to change them.
- The **certVolumeClaimTemplate** is used by the nginx pod and in 2DCDR mode it is also by the **tunnel**, **db-remote** and **www-remote** pods. It can be sized small, just like the **adminVolumeClaimTemplate**, and does not need to be increased based on the number of sites.

$LICENSE_ACCEPTANCE
: Set **accept** to **true**. You must accept the license to successfully deploy API Connect.

$LICENSE_USE
: Set **use** to either **production** or **nonproduction** to match the license that you purchased.

$LICENSE_ID
: Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

4. Optional: If you want to disable mTLS for communications between the management and portal, and enable JWT instead, then add and set the properties **mtlsValidateClient** and **jwksUrl**.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the **jwksUrl**, describe the management CR and check the **status:** section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.

5. Install the Portal subsystem by applying the modified CR with the following command:

```
oc apply -f portal_cr.yaml -n <namespace>
```

6. Verify that the Portal subsystem is fully installed:

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

The installation is complete when **STATUS** reports **Running**, **Ready** reports that all services are online (**3/3**), and **MESSAGE** reports "Ready for API Cloud Manager service registration", for example:

```
NAME           READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE                                    AGE
portal         3/3     Running   <version>  <version>            Ready for API Cloud Manager service registration   2h
```

You do not need to wait for the portal installation to complete before you move on to the next subsystem installation.

7. Verify that you can access the portal web endpoint by browsing to the **portalUIEndpoint.hosts.name** as configured in your yaml file:

```
curl -k https://portal.$STACK_HOST
<!DOCTYPE html>
<html>
<head>
<title>Developer Portal is ready to create sites!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>Developer Portal is ready to create sites!</h1>
<p>If you see this page, the web server is successfully installed and
working. Further configuration is required.</p>
</body>
</html>
```

8. It is highly recommend that you retain the Portal subsystem encryption secret in a safe location, in case you need to restore the Portal subsystem during a disaster recovery scenario.
The encryption secret is created when the Portal subsystem is initially deployed. You can obtain the name of the encryption secret from the Portal subsystem custom resource after the installation is complete:

    a. Obtain the name of the encryptionSecret:

```
oc get ptl portal -o yaml -n <namespace> | grep encryptionSecret
encryptionSecret: portal-enc-key
```

The name of encryption secret is **portal-enc-key**.

b. Save the secret yaml in a safe location:

```
oc get secret portal-enc-key -o yaml -n <namespace> > portal-enc-key.yaml
```

9. Backup the Portal installation. See Backing up and restoring the developer portal.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing the Gateway subsystem
Next topic: Installing the Analytics subsystem

Installing the Analytics subsystem

Install the Analytics subsystem by creating and applying the ingress-ca, common-issuer, and analytics_cr yaml files.

Before you begin

When installing subsystems in different namespaces you must first create the cert-manager issuers and certificates required by that subsystem in the same namespace.

1. Using the file obtained from Extracting the Management ingress-ca certificates, run: `oc apply -f ingress-ca.yaml -n <analytics namespace>`
2. Create a file called **common-issuer.yaml** and paste in these contents:
   Note: If you have just done Installing the Portal subsystem then you will already have this file.

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
```

3. Apply the above yaml file with: `oc apply -f common-issuer.yaml -n <analytics namespace>`
4. Confirm that the issuers were created and are in ready state: `oc get issuers -n <analytics namespace>`

```
oc get issuers -n <analytics namespace>
NAME                 READY
ingress-issuer       True
selfsigning-issuer   True
```

About this task

Use the OpenShift CLI to edit the custom resource template for the Analytics subsystem, apply the resource, verify that the pods are up and running.

Procedure

1. Create a file that is called **analytics_cr.yaml** and paste in these contents:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsCluster
metadata:
  name: analytics
  labels: {
    app.kubernetes.io/instance: "analytics",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics"
  }
spec:
  version: $APP_PRODUCT_VERSION
  license:
    accept: $LICENSE_ACCEPTANCE
    use: $LICENSE_USE
    license: '$LICENSE_ID'
  profile: $PROFILE
  microServiceSecurity: certManager
  certManagerIssuer:
    name: selfsigning-issuer
    kind: Issuer
  ingestion:
    endpoint:
      annotations:
        cert-manager.io/issuer: ingress-issuer
      hosts:
      - name: ai.$STACK_HOST
        secretName: analytics-ai-endpoint
    clientSubjectDN: CN=analytics-ingestion-client,O=cert-manager
  storage:
    type: shared # change to 'dedicated' if you want separate master and storage pods. Three-replica deployments only.
    shared:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
        volumeSize: $DATA_VOLUME_SIZE
    # master: # uncomment this section if you set storage.type = dedicated.
    #   volumeClaimTemplate:
    #     storageClassName: $STORAGE_CLASS
```

2. Edit the YAML file and set the variables:

$APP_PRODUCT_VERSION
API Connect application version for the subsystems.

version: <*version_number*>

Example version number: 10.0.7.0

$PROFILE
Specify your analytics subsystem profile, where **n** indicates number of replicas, **c** number of cores, and **m** is the minimum memory allocation in GB. For more information on profiles, see Deployment and component profiles.

$STACK_HOST
The desired ingress subdomain for the API Connect stack. Used when specifying endpoints. Domain names that are used for endpoints cannot contain the underscore "_" character. You can customize the subdomain or the complete hostname:

- Subdomain customization only
  Accept the prefixes predefined for the ingress hostnames to use and replace all instances of **$STACK_HOST** with the desired ingress subdomain for the API Connect stack. For example, if your host is **myhost.subnet.example.com**:

```
  ingestion:
    endpoint:
      < ... >
    hosts:
    - name: ai.myhost.subnet.example.com
      secret: analytics-ai-endpoint
     < .... >
```

- Complete hostname customization

Change both the predefined prefixes and the **$STACK_HOST** subdomain to match your desired hostnames.

For example, you can replace **$STACK_HOST** with **my.analytics.ingestion.myhost.subnet.example.com**, where **my.analytics.ingestion** replaces **ai**, and **myhost.subnet.example.com** replaces **STACK_HOST**.

```
ingestion:
    endpoint:
      < ... >
      hosts:
      - name: my.analytics.ingestion.myhost.subnet.example.com
        secret: analytics-ai-endpoint
       < ... >
```

$STORAGE_CLASS
The Kubernetes storage class to be used for persistent volume claims. For more information, see Analytics preinstallation planning. Find the available storage classes in the target cluster by running the following command: **oc get sc**. Example:

```
storage:
    type: shared
    shared:
      volumeClaimTemplate:
        storageClassName: ceph-block
```

$DATA_VOLUME_SIZE
Size of storage allocated for data. To estimate the storage space you require, see Estimating storage requirements. If you are unable to estimate your storage requirement then set it to 500Gi.

```
storage:
    type: shared
    shared:
      volumeClaimTemplate:
        storageClassName: ceph-block
        volumeSize: 500Gi
```

$LICENSE_ACCEPTANCE
Set **accept** to **true**. You must accept the license to successfully deploy API Connect.
$LICENSE_USE
Set **use** to either **production** or **nonproduction** to match the license that you purchased.
$LICENSE_ID
Set **license:** to the license ID for the version of API Connect that you purchased. See API Connect licenses.

3. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the **YAML** tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

4. Optional: Three replica deployments only: Enable dedicated storage.
If you want to use dedicated storage, update the **spec.storage** section of the yaml file as follows:

```
  storage:
    type: dedicated
    shared:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
        volumeSize: $DATA_VOLUME_SIZE
    master:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
```

For more information about dedicated storage, see dedicated or shared storage.
5. Optional: If you want to disable mTLS for communications between the management and analytics subsystem, and between the gateway and analytics subsystem, and enable JWT instead, then add and set the properties **mtlsValidateClient** and **jwksUrl**.

```
spec:
  ...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

where *<JWKS URL>* is the URL of the JWKS endpoint that is hosted on the management subsystems. To find out the **jwksUrl**, describe the management CR and check the **status:** section:

```
kubectl describe mgmt -n <namespace>
...
status:
  - name: jwksUrl
    secretName: api-endpoint
    type: API
    uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

For more information about JWT security, see Enable JWT security instead of mTLS.
Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow.
6. Install the analytics subsystem by applying the modified CR with the following command:

```
oc apply -f analytics_cr.yaml -n <namespace>
```

7. Verify that the analytics subsystem is fully installed:

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

The installation is complete when **READY** shows all pods running (**n/n**), and the **STATUS** reports **Running**. Example:

```
NAME         READY   STATUS    VERSION          RECONCILED VERSION   AGE
analytics    n/n     Running   10.0.7.0   10.0.7.0-1281    86m
```

It is not necessary to wait for analytics installation to complete before you move on to the next subsystem installation.

8. Backup your analytics subsystem and configure scheduled analytics database backups: Backing up and restoring the analytics subsystem.

What to do next

If you are creating a new deployment of API Connect, install other subsystems as needed.

When you have completed the installation of all required API Connect subsystems, you can proceed to defining your API Connect configuration by using the API Connect Cloud Manager; refer to the Cloud Manager configuration checklist.

Previous topic: Installing the Portal subsystem

Installing with the top-level CR on OpenShift

Use the single, top-level **APIConnectCluster** CR to deploy API Connect on OpenShift.

About this task

When you deploy API Connect with a top-level CR, all the API Connect subsystems are deployed in the same namespace. The CR contains a section for each subsystem; configure deployment settings in the appropriate section for each subsystem.
Note: The top-level CR installation creates and manages one instance of each subsystem. If you want multiple portal, gateway, or analytics subsystems in your API Connect deployment, then install the API Connect subsystems individually: Installing API Connect in a single namespace on OpenShift. For more information about API Connect installation options on OpenShift, see OpenShift: Deciding to use individual CRs, a top-level CR, or Cloud Pak for Integration.
If you want to install multiple API Connect instances, for example dev, uat, and prod, the recommended configuration is to have single API Connect and Datapower operators installed globally in the **openshift-operators** namespace and each API Connect instance installed in its own namespace using a top-level CR.

Attention: If you want to configure FIPS support for API Connect, you must enable FIPS on the cluster as part of the initial OpenShift deployment. You cannot modify an existing cluster to enable FIPS support. For more information, see Configuring FIPS support on OpenShift.

- **Installing operators**
  Install operators for API Connect, DataPower, and IBM Cloud Pak foundational services so that you can deploy API Connect on Red Hat OpenShift.
- **Installing API Connect**
  Use the top-level **APIConnectCluster** CR to install all the API Connect subsystems into a shared namespace on OpenShift while connected to the internet.

Installing operators

Install operators for API Connect, DataPower, and IBM Cloud Pak foundational services so that you can deploy API Connect on Red Hat OpenShift.

Before you begin

The API Connect operator and operand must be from the same release and fix pack level. Table 1 lists the current version of the operator and operand for API Connect.

Table 1. API Connect operator and operand versions

API Connect (operand)	Operator channel version	Highest operator version
10.0.7.0	v5.1	5.1.0

Note: There can be only one instance of the API Connect operator in your OpenShift environment. You cannot have multiple namespaces each with their own API Connect operator.
You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

Procedure

1. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step 3.
   The **ibm-pak** plug-in enables you to access hosted product images, and to run **oc ibm-pak** commands against the cluster. To confirm that **ibm-pak** is installed, run the following command and verify that the response lists the command usage:

   ```
   oc ibm-pak --help
   ```

2. Obtain an entitlement key for the Entitled Registry:
   Air-gapped: If you are in an air-gapped environment, then skip this step and instead follow the steps in Air-gapped installation, and then continue from step #inst_ocp_cp4i_ops__d322e179.
   a. Log in to the IBM Container Library.
   b. In the Container software library, select Get entitlement key.
   c. After the Access your container software heading, click Copy key.
   d. Copy the key to a safe location.
3. Create the namespace where API Connect will be installed, either by selecting Home > Projects > Create Project in the UI, or with the following command:

   ```
   oc create ns <APIC-namespace>
   ```

The namespace where you install API Connect must meet the following requirements:

- Red Hat OpenShift: Only one top-level CR (`APIConnectCluster`) can be deployed in each namespace.
- Cloud Pak for Integration: Only one API Connect capability can be deployed in each namespace.
- Red Hat OpenShift restricts the use of default namespaces for installing non-cluster services. The following namespaces cannot be used to install API Connect:
  - `default`
  - `kube-system`
  - `kube-public`
  - `openshift-node`
  - `openshift-infra`
  - `openshift`

4. If you are installing the operator into a single namespace, create an OperatorGroup object specifying that namespace.
   Attention: If you are installing the operator into all namespaces, skip this step because the `openshift-operators` namespace already has an appropriate Operator group in place.
   a. Create the apiconnect-operator-group.yaml file with an OperatorGroup object that identifies the namespace in which RBAC permissions for the IBM API Connect Operator will be generated, and where the CSV will be available:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operator-group-name>
  namespace: <APIC-namespace>
spec:
  targetNamespaces:
  - <APIC-namespace>
```

   b. Apply the OperatorGroup object with the following command:

```
oc apply -f apiconnect-operator-group.yaml
```

5. If you are installing the operator in all namespaces, then create the common services namespace:

```
oc create ns ibm-common-services
```

6. Create a pull secret in the namespaces where you want to install API Connect.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   a. Open the Red Hat OpenShift web console, click Workloads > Secrets.
   b. Ensure that the Project is set to the namespace where you intend to install API Connect.
   c. Click Create and select Image pull secret.
   d. Set the following parameters for the secret:
      - Set Secret name to `ibm-entitlement-key`.
      - Set Authentication type to `Image registry credentials`.
      - Set Registry server address to `cp.icr.io`.
      - Set Username to `cp`.
      - Set Password to the entitlement key generated in Step 2.
      - Click Create to create the secret.

   Note: If your Red Hat OpenShift platform is Red Hat OpenShift on IBM Cloud (ROKS), you might need to reload your worker nodes after the image pull secret is created. You can reload the worker nodes either from the Red Hat OpenShift web console or from the command line with the following command: `ibmcloud oc worker reload`
   For information on using the `reload` command, see the Red Hat OpenShift on IBM Cloud documentation.

7. Add the API Connect catalog sources to your cluster.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   Adding catalog sources to your Red Hat OpenShift cluster adds the IBM operators to the list of operators you can install. For most connected and air-gapped clusters, applying individual catalog sources is the most effective way to fully control software versioning on the cluster.

   a. Generate the catalog source files:
      i. Log into your cluster using the `oc login` command and your user credentials:

```
oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
```

      ii. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64
```

      For information on API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

      iii. Download the files for the operators required by API Connect:

```
oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
```

      iv. Generate the catalog sources for API Connect:

```
oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources (run both commands):

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

   b. Apply the catalog sources:
      i. Apply the catalog sources (run both commands):

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

ii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

8. Add the IBM Cloud Pak foundational services (common services) catalog sources to your cluster.
   Air-gapped: If you are in an air-gapped environment, then skip this step as it is covered in Air-gapped installation.
   a. Generate the catalog source files:
      i. Log into your cluster using the `oc login` command and your user credentials:

```
oc login <openshift_url> -u <username> -p <password> -n <APIC-namespace>
```

      ii. Export the variables for the command line to use; substituting the appropriate CASE version:

```
export CASE_NAME=ibm-cp-common-services
export CASE_VERSION=<case version>
export ARCH=amd64
```

      Set *<case version>* to the value specified for *Foundational services channel version* in Operator, CASE, and platform requirements, for your deployment type.

      iii. Download the files for the operators required by IBM Cloud Pak foundational services (common services):

```
oc ibm-pak get ${CASE_NAME} --version ${CASE_VERSION}
```

      iv. Generate the catalog sources:

```
oc ibm-pak generate mirror-manifests ${CASE_NAME} icr.io --version ${CASE_VERSION}
```

      v. Optionally get the catalog sources and save them in another directory in case you need to replicate this installation in the future:
      Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

      This command might not return anything if there are no architecture-specific catalog sources.

   b. Apply the catalog sources:
      i. Apply the catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

      ii. Confirm that the catalog sources have been created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

9. Create the `apiconnect` subscription.
   a. Create the `apiconnect` subscription with the appropriate channel by creating a file called `apic-sub.yaml` and pasting in the following contents, updating the namespace as required:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-apiconnect
  namespace: <namespace>
spec:
  channel: v5.1
  name: ibm-apiconnect
  source: ibm-apiconnect-catalog
  sourceNamespace: openshift-marketplace
```

   Where `<namespace>` is one of the following values:
   - `openshift-operators` if you are installing the operator in all namespaces
   - The namespace created in step 3 if you are installing the operator in a single namespace

   b. Apply the subscription with the following command:

```
oc apply -f apic-sub.yaml
```

   c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
   If any operators such as `ibm-apiconnect` or `IBM DataPower Gateway` show the status of "Upgrade available", approve the upgrade by completing the following steps:

      i. Click Upgrade available.
      ii. Click Preview InstallPlan.
      iii. Click Approve.
      iv. Wait for the `IBM API Connect` and `IBM DataPower Gateway` operators to install.

   The `IBM DataPower Gateway` operator is a prerequisite to API Connect and must not be removed.

10. Install cert manager by completing the following steps:
    a. Log in to the OpenShift Container Platform web console.

    b. Navigate to Operators > OperatorHub.

    c. In the filter box, type: `cert-manager Operator for Red Hat OpenShift`.

    d. Select cert-manager Operator for Red Hat OpenShift and click Install.

    e. On the Install Operator page, complete the following steps:

i. Update the Update channel if needed. The channel defaults to **stable-v1**, which installs the latest stable release of the cert-manager Operator for Red Hat OpenShift.

ii. Select the Installed Namespace for the operator.
The default operator namespace is **cert-manager-operator**; if that namespace doesn't exist, it is created for you.

iii. Select an Update approval strategy:
- Automatic: allow Operator Lifecycle Manager (OLM) to automatically update the operator when a new version is available.
- Manual: require a user with the appropriate credentials to approve all operator updates.

iv. Click Install.

f. Verify the cert-manager installation by completing the following steps:

i. Click to Operators > Installed Operators.

ii. Verify that **cert-manager Operator for Red Hat OpenShift** is listed with a Status of **Succeeded** in the **cert-manager-operator** namespace.

iii. Verify that cert-manager pods are up and running by running the following command:

```
oc get pods -n cert-manager
```

For a successful installation, the response looks like the following example:

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-bd7fbb9fc-wvbbt               1/1     Running   0          3m39s
cert-manager-cainjector-56cc5f9868-7g9z7   1/1     Running   0          4m5s
cert-manager-webhook-d4f79d7f7-9dg9w       1/1     Running   0          4m9s
```

11. Create the **ibm-common-services-operator** subscription, if it does not already exist.

a. Create a file called common-services-sub.yaml and paste in the following contents:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-common-service-operator
  namespace: <namespace>
spec:
  channel: v4.3
  name: ibm-common-service-operator
  source: opencloud-operators
  sourceNamespace: openshift-marketplace
```

Where: ***<namespace>*** is one of the following values:

- **openshift-operators** if you are installing the operator in all namespaces
- The namespace created in step 3 if you are installing the operator in a single namespace

b. Apply the subscription with the following command:

```
oc apply -f common-services-sub.yaml
```

c. Select Operators > Installed Operators, and ensure that Project: All Projects is selected.
If any operators such as **ibm-apiconnect**, **IBM Cloud Pak foundational services**, or **Operand Deployment Lifecycle Manager** show the status of "Upgrade available", approve the upgrade by completing the following steps:

i. Click Upgrade available.
ii. Click Preview InstallPlan.
iii. Click Approve.
iv. Check the **ibm-common-services** namespace and ensure that all operators with a status of "Upgrade available" are approved.
v. Wait for the **IBM Cloud Pak foundational services** and **Operand Deployment Lifecycle Manager** operators to install.

Installing API Connect

Use the top-level **APIConnectCluster** CR to install all the API Connect subsystems into a shared namespace on OpenShift while connected to the internet.

Before you begin

Review installation considerations and complete and prerequisite tasks as explained in:

1. Preparing for installation
2. Installing operators

Important: When you install API Connect, you must configure secure communication between subsystems. Use mTLS or JWT for management to portal and analytics subsystems. Use JWT for management to gateway communication. For more information on inter-subsystem communication, see Network requirements for inter-subsystem communication.

Procedure

1. Install the subsystems using the OpenShift web console.
API Connect uses a single top-level **APIConnectCluster** CR to quickly deploy all of the subsystems. With the **APIConnectCluster** CR, you provide values for a few properties, and then the installation logic chooses sensible defaults for the other properties during subsystem deployment.

a. Click Operators->Installed Operators.

b. Select the IBM API Connect operator.
c. Click Create Instance on the API Connect cluster tile to install the top-level Custom Resource.
Selecting API Connect cluster installs all the API Connect subsystems with the default configuration. The API Connect subsystems are Management, Developer Portal, Gateway, and Analytics.

d. Provide values for the fields shown in Table 1.

Table 1. Properties for the API Connect top-level CR

Input	Value
Name	The name to be used when deploying the `APIConnectCluster` CR.
License acceptance	Required. Click to accept the license. You must accept the license to install API Connect.
License use	Required. Select production or nonproduction to match the type of license that you purchased.
License metric	Optional. Enter the unit of measurement that is used for your program license: • PROCESSOR_VALUE_UNIT - Default value. If you leave the field blank, this value is used. • MONTHLY_API_CALL - Applies only to the IBM API Connect Hybrid Entitlement program.
License ID	Required. Select the license ID for the API Connect program that you purchased. To view all license IDs, see API Connect licenses.
Deployment profile	The deployment profile defines the number of worker nodes, CPU, and memory that is available for API Connect. The available profiles are listed in API Connect deployment profiles for OpenShift and Cloud Pak for Integration.
Product version	Optional. The API Connect application version. Defaults to the latest version available.
Default storage class name	The name of the Storage class to be used. You can set a default to appear here with the command: `oc patch storageclass <storage class name> -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class":"true"}}}'`
Database Backup settings	
Backup Certs	The name of the Kubernetes secret containing your S3 Access Key / Key Secret. Default: (Generated Value) See Configuring management database backups.
Backup S3 URI style	Indicates whether URIs to your S3 backup specify a host or a path value. Set to either `host` or `path`.
Credentials	The name of the Kubernetes secret containing your S3 Access Key / Key Secret. Default: (Generated Value) See Configuring management database backups.
Server hostname	The backups host. Specify S3 endpoint with the corresponding S3 region in the format `<S3endpoint>/<S3region>`. For example: `s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard` Default: none See Configuring management database backups.
Path	The path to the location of the backup. The name of your S3 bucket to store backup data. For example, `bucket-name/folder`. The use of subdirectories in the bucket name is not supported. Ensure that `bucket-name/folder` is empty. Default: none See Configuring management database backups.
Server port	Not used for S3 object-store backups.
RepoRetentionFull	The number of days to retain backups for. For example, if `RepoRetentionFull` is set to 30, then any backups older than 30 days are deleted.
Schedule	Cron like schedule for performing automatic backups. The format for the schedule is: • ****** • ------ • |||||| • ||||| +----- day of week (0 - 6) (Sunday=0) • |||| +------- month (1 - 12) • ||| +--------- day of month (1 - 31) • || +----------- hour (0 - 23) • | +------------- min (0 - 59) • +--------------- sec (0 - 59) The backup schedule defaults to `0 0 * * * *`. This means a backup is run every day at midnight and minute zero UTC. When you configure a host, if you do not specify a value for `schedule`, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed. See Configuring management database backups.

Note: If you do not have an S3 object-store available, you can install API Connect with no backup settings configured, by clicking the YAML tab and manually adding the following annotation:

```
metadata:
  annotations:
    apiconnect-operator/backups-not-configured: "true"
```

If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
e. If you want to configure additional settings, click Advanced and see API Connect configuration settings for information on the advanced settings.
f. Optional: If Cloud Pak for Integration is installed, and you do not want your API Connect installation to be integrated with it, then click the `YAML` tab and manually add the following annotation to the metadata section of the CR:

```
metadata:
  annotations:
    apiconnect-operator/cp4i: "false"
```

This annotation disables all integration with Cloud Pak for Integration.

2. Optional: If you are installing API Connect as a Cloud Pak for Integration component and you want to deploy with Cloud Pak endpoints, complete the following steps. This step is not applicable if you already disabled API Connect integration with Cloud Pak for Integration by setting `metadata.annotations.apiconnect-operator/cp4i` to false.
   Beginning with version 10.0.7.0, API Connect no longer uses the Cloud Pak cpd routes for endpoints when deployed as a component of Cloud Pak for Integration. Instead, the API Management component uses the typical default API Connect routes (or the custom endpoints configured in the CR).

   a. Click the YAML tab to manually edit the CR.
   b. Insert the `deprecatedCloudPakRoute` object into the `spec.management` section as shown in the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  name: <name_of_your_instance>
  namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-MMBZ-295QZQ
    metric: PROCESSOR_VALUE_UNIT
    use: nonproduction
  profile: n1xc17.m48
  version: 10.0.7.0
  storageClassName: <default-storage-class>
  management:
    deprecatedCloudPakRoute:
      enabled: true
      cloudPakEndpoint:
        hosts:
        - name: cpd-apic.apps.admin-<domain>.com
```

   c. Optional: If you want to add a custom certificate for the Cloud Pak route, complete the following steps:
      i. Create a Cloud Pak certificate that was signed by a Cloud Pak CA as in the following example:

```
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
 name: custom-cpd-ca
 namespace: apic
spec:
 selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: custom-cpd
  namespace: apic
spec:
  commonName: small-mgmt-cpd
  dnsNames:
  - cpd-apic.apps.admin-apickeycloak.cp.fyre.ibm.com
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: custom-cpd-ca
  privateKey:
    algorithm: RSA
    rotationPolicy: Always
    size: 2048
  renewBefore: 720h0m0s
  secretName: custom-cpd
  usages:
  - key encipherment
  - digital signature
  - server auth
```

      ii. In the CR, provide the secret name within the `cloudPakEndpoint` property of the new `deprecatedCloudPakRoute` object; for example:

```
spec:
  deprecatedCloudPakRoute:
    enabled: true
    cloudPakEndpoint:
      hosts:
      - name: cpd-apic.apps.admin-<domain>.com
        secretName: custom-cpd
```

3. Click Create to install API Connect.
4. To verify that your API Connect cluster is successfully installed, run `oc get apic -n <APIC-namespace>`.
5. Verify that you can log in to the API Connect Cloud Manager UI:
   a. To determine the location for logging in, view all the endpoints:

```
oc get routes -n <APIC-namespace>
```

   b. Locate the `mgmt-admin-apic` endpoint, and access the Cloud Manager UI.
   c. Login as `admin`.

When you install with the top-level CR, the password is auto-generated. To get the password, run the following commands:

```
oc get secret -n <APIC-namespace> | grep mgmt-admin-pass

oc get secret -n <APIC-namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -d && echo ""
```

6. Optional: Complete the following tasks:
   - Increase the timeout settings for management endpoints, particularly for large deployments. See Configuring timeouts for management endpoints.
   - Download the API Connect toolkit from the Cloud Manager UI or API Manager UI after installation is completed. See Installing the toolkit.
   - Install a test environment. See Testing an API with the Local Test Environment.

   Note: You can also install the Toolkit and Local Test Environment from IBM Fix Central. For a link to the latest files on Fix Central, see What's new in the latest release.

What to do next

Configure backups for all your API Connect subsystems: Backing up, restoring, and disaster recovery.

Air-gapped installation

In a disconnected environment using Red Hat OpenShift Container Platform (OCP), install API Connect by mirroring product images to a bastion host or a portable device and then completing the installation.

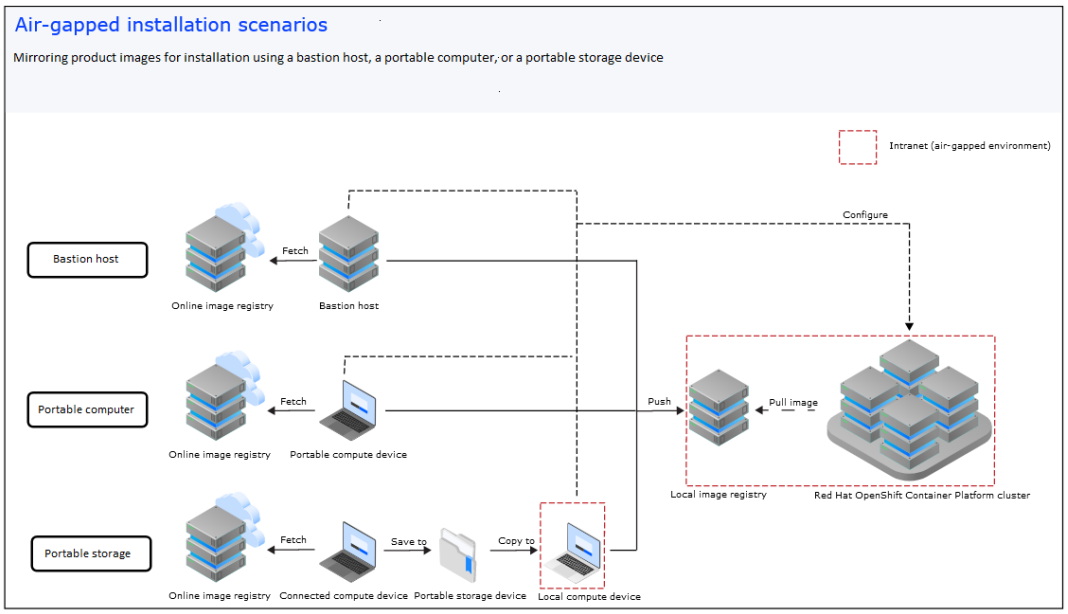
Before you begin

- An air-gapped installation is not possible in the following environments:
  - Red Hat OpenShift Service on AWS (ROSA)
  - Red Hat OpenShift Service Kubernetes Service on IBM Cloud (ROKS)

About this task

If your cluster is not connected to the internet (it is network-restricted, or air-gapped), you can install API Connect in your cluster by using either a bastion host, a portable computer, or a portable storage device.

The following diagram illustrates how you can mirror product images for an air-gapped installation with a bastion host, a portable computer, or a portable storage device.



All of the scenarios use Container Application Software for Enterprises (CASE) files to mirror content from a source (on the internet) to an intermediary target (a bastion host, a portable computer, or a portable storage device) before pushing them to a local server or device.

Then, you can transfer the files to the local air-gapped network and run the installation.

- **Installing with a bastion host**
  You can use a bastion host to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.
- **Installing with a portable computer or storage device**
  You can use a portable device, such as a portal computer or a USB storage device, to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Installing with a bastion host

You can use a bastion host to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Before you begin

This task must be performed by a Red Hat OpenShift administrator.

About this task

If your cluster is not connected to the internet, you can mirror product images to a registry in your network-restricted environment by using a bastion host. A bastion host has access to both the public internet and the network-restricted environment where the target clusters reside. You can fetch product images from the internet and push them to a local registry; then you can pull the images from the local registry to the target cluster for installation.

Procedure

1. Set up the mirroring environment.
   a. Prepare the target cluster:
      - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
        For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in IBM API Connect Version 10 software product compatibility requirements.
      - Configure storage on the cluster and make sure that it is available.

   b. Prepare the bastion host:
      You must be able to connect your bastion host to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry) at the same time. Your host must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

      i. Ensure that the sites and ports listed in Table 1 can be reached from the bastion host:

      Table 1. Sites that must be reached from the bastion host

Site	Description
`icr.io:443`	IBM entitled registry
`quay.io:443`	Local API Connect image repository
`github.com`	CASE files and tools
`redhat.com`	Red Hat OpenShift Container Platform (OCP) upgrades

      ii. On the bastion host, install either Docker or Podman (not both).
      Docker and Podman are used for managing containers; you only need to install one of these applications.

      - To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

        ```
        yum check-update
        yum install docker
        ```

      - To install Podman, see the Podman installation instructions.
        For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

        ```
        yum install podman
        ```

      iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in Getting started with the OpenShift CLI.
      The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

      iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
      The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

      ```
      oc ibm-pak --help
      ```

   c. Set up a local image registry and credentials.
      The local Docker registry stores the mirrored images in your network-restricted environment.

      i. Install a registry, or get access to an existing registry.
      You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

      The registry product that you use must meet the following requirements:
      - Supports multi-architecture images through Docker Manifest V2, Schema 2
        For details, see Docker Manifest V2, Schema 2.
      - Is accessible from the Red Hat OpenShift Container Platform cluster nodes
      - Allows path separators in the image name

      Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

      ii. Configure the registry to meet the following requirements:
      - Supports auto-repository creation
      - Has sufficient storage to hold all of the software that is to be transferred
      - Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
      - Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

      To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

2. Mirror the cert-manager operator for Red Hat OpenShift.
   Mirror the operator as explained in the Red Hat instructions: Mirroring images for a disconnected installation using the oc-mirror plugin .

When you modify the **ImageSetConfiguration**, include **openshift-cert-manager-operator** at channel **stable-v1** as one of the packages, as shown in the following example:

```
packages:
- name: openshift-cert-manager-operator
  channels:
  - name: stable-v1
```

3. Set environment variables and download CASE files.
   Because you will use values from two different CASE files, you must create environment variables for both; notice that the variables for the foundational services (common services) CASE file are prefixed with "CS_" to differentiate them.

   a. Create the following environment variables with the installer image name and the image inventory on your host:
      - API Connect:

        ```
        export CASE_NAME=ibm-apiconnect
        export CASE_VERSION=5.1.0
        export ARCH=amd64
        ```

        For information on API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

      - Foundational services:

        ```
        export CS_CASE_NAME=ibm-cp-common-services
        export CS_CASE_VERSION=1.15.10
        export CS_ARCH=amd64
        ```

        For information on IBM Cloud Pak foundational services (common services) CASE versions, see "Table 1. Image versions for offline installation" in Installing IBM Cloud Pak foundational services in an air-gapped environment in the IBM Cloud Pak foundational services documentation.

      - EDB:

        ```
        export EDB_CASE_NAME=ibm-cloud-native-postgresql
        export EDB_CASE_VERSION=4.18.0
        ```

   b. Connect your host to the internet (it does not need to be connected to the network-restricted environment at this time).
   c. Download the CASE files to your portable device:
      - API Connect:

        ```
        oc ibm-pak get $CASE_NAME --version $CASE_VERSION
        ```

      - Foundational services:

        ```
        oc ibm-pak get $CS_CASE_NAME --version $CS_CASE_VERSION
        ```

      - EDB:

        ```
        oc ibm-pak get $EDB_CASE_NAME --version $EDB_CASE_VERSION
        ```

4. Mirror the images.
   The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

   a. Generate mirror manifests.
      i. Define the environment variable $TARGET_REGISTRY by running the following command:

         ```
         export TARGET_REGISTRY=<target-registry>
         ```

         Replace ***<target-registry>*** with the IP address (or host name) and port of the local registry; for example: **172.16.0.10:5000**. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: **172.16.0.10:5000/registry_ns**.
      ii. Generate mirror manifests by running the following commands:
         - API Connect:

           ```
           oc ibm-pak generate mirror-manifests $CASE_NAME --version $CASE_VERSION $TARGET_REGISTRY
           ```

         - Foundational services:

           ```
           oc ibm-pak generate mirror-manifests $CS_CASE_NAME --version $CS_CASE_VERSION $TARGET_REGISTRY
           ```

         - EDB:

           ```
           oc ibm-pak generate mirror-manifests $EDB_CASE_NAME --version $EDB_CASE_VERSION $TARGET_REGISTRY
           ```

         If you need to filter for a specific image group, add the parameter **--filter** ***<image_group>*** to the command.

   The **generate** command creates the following files at ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION, ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION, and ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION:
   - catalog-sources.yaml
   - catalog-sources-linux-*<arch>*.yaml (if there are architecture-specific catalog sources)
   - image-content-source-policy.yaml
   - images-mapping.txt

   The files are used when mirroring the images to the **TARGET_REGISTRY**.

   b. Obtain an entitlement key for the entitled registry where the images are hosted:
      i. Log in to the IBM Container Library.
      ii. In the Container software library, select Get entitlement key.
      iii. In the "Access your container software" section, click Copy key.
      iv. Copy the key to a safe location; you will use it to log in to **cp.icr.io** in the next step.
   c. Authenticate with the entitled registry where the images are hosted.

The image pull secret allows you to authenticate with the entitled registry and access product images.

i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at $HOME/.docker/config.json on Linux or %USERPROFILE%/.docker/config.json on Windows.

ii. Log in to the `cp.icr.io` registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use `cp` as the username and your entitlement key as the password.

d. Authenticate with the local registry.
Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

e. Update the CASE manifest to correctly reference the DataPower Operator image.
Files for the DataPower Operator are now hosted on `icr.io`; however, the CASE manifest still refers to `docker.io` as the image host. To work around this issue, visit Airgap install failure due to 'unable to retrieve source image docker.io' in the DataPower documentation and update the manifest as instructed. After the manifest is updated, continue to the next step in this procedure.

f. Mirror the product images.

i. Connect the bastion host to both the internet and the restricted-network environment that contains the local registry.

ii. Run the following commands to copy the images to the local registry:

- API Connect:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \
  -a $REGISTRY_AUTH_FILE \
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1
```

- Foundational services:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping.txt \
  -a $REGISTRY_AUTH_FILE \
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1
```

- EDB:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping.txt \
  -a $REGISTRY_AUTH_FILE \
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the `--insecure` option to the command.
There might be a slight delay before you see a response to the command.

g. Configure the target cluster.
Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

ii. Update the global image pull secret for the cluster as explained in the Red Hat OpenShift Container Platform documentation.
Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the TARGET_REGISTRY to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:
  registrySources:
    insecureRegistries:
    - insecure0.svc:5001
    - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

iii. Create the ImageContentSourcePolicy, which instructs the cluster to pull the images from your local registry:

- API Connect:

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

- Foundational services:

```
oc apply -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/image-content-source-policy.yaml
```

- EDB:

```
oc apply -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/image-content-source-policy.yaml
```

iv. Verify that the ImageContentSourcePolicy resource was created:

```
oc get imageContentSourcePolicy
```

v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

5. Apply the catalog sources.
Now that you have mirrored images to the target cluster, apply the catalog sources for API Connect and Foundational services. There is no need to apply the catalog source for EDB because the API Connect operator will apply that catalog source in its own namespace when the API Connect operand is created.

In the following steps, replace ***<Architecture>*** with either **amd64**, **s390x** or **ppc64le** as appropriate for your environment.

a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64

export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=5.0.0
export CS_ARCH=amd64
```

b. Generate the catalog sources and save them in another directory in case you need to replicate this installation in the future.
   i. Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

cat ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

   ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

c. Apply the catalog sources to the cluster.
   i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

oc apply -f ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

   ii. Apply any architecture-specific catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

   iii. Confirm that the catalog sources were created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

What to do next

Install API Connect following the steps for your deployment type: Deploying on OpenShift and Cloud Pak for Integration.

Installing with a portable computer or storage device

You can use a portable device, such as a portal computer or a USB storage device, to perform an air-gapped installation of API Connect on Red Hat OpenShift Container Platform (OCP) when your cluster has no internet connectivity.

Before you begin

This task must be performed by a Red Hat OpenShift administrator.

About this task

If your cluster is not connected to the internet, you can mirror product images to a registry in your network-restricted environment by using a portable device. Download images from the public registries on the internet to the portable device; then bring that portable device into the network-restricted environment (or transfer the files to a different device within the network-restricted environment) and mirror the images to the local registry. You can pull the images from the local registry to the target cluster for installation.

Procedure

1. Set up the mirroring environment.
   a. Prepare the target cluster:
      - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
        For more information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in Operator, CASE, and platform requirements.

- Configure storage on the cluster and make sure that it is available.

b. Prepare the portable device:
You must be able to connect your portable device to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry). The portable device must be on a Linux x86_64 or Mac platform with an operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

i. Check that the portable device has sufficient storage to hold all the software that is to be transferred to the local registry.
ii. On the portable device, install either Docker or Podman (not both).
Docker and Podman are used for managing containers; you can install either of these applications.

- To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

```
yum check-update
yum install docker
```

- To install Podman, see the Podman installation instructions.
For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

```
yum install podman
```

iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in Getting started with the OpenShift CLI.
The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

c. Set up a local image registry and credentials.
The local Docker registry stores the mirrored images in your network-restricted environment.

i. Install a registry, or get access to an existing registry.
You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:
- Supports multi-architecture images through Docker Manifest V2, Schema 2.
For more information, see Docker Manifest V2, Schema 2.
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes.
- Allows path separators in the image name.

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

ii. Configure the registry to meet the following requirements:
- Supports auto-repository creation.
- Has sufficient storage to hold all the software that is to be transferred.
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials).
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials).

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries at run time, use an account that can read from the target local registry.

2. Mirror the cert-manager operator for Red Hat OpenShift.
Mirror the operator as explained in the Red Hat instructions: Mirroring images for a disconnected installation using the oc-mirror plugin .

When you modify the `ImageSetConfiguration`, include `openshift-cert-manager-operator` at channel `stable-v1` as one of the packages, as shown in the following example:

```
  packages:
  - name: openshift-cert-manager-operator
    channels:
    - name: stable-v1
```

3. Set environment variables and download CASE files.
Create environment variables to use while mirroring images, connect to the internet, and download the CASE files needed for installing API Connect.

a. Create the following environment variables with the installer image name and the image inventory on your portable device:
- API Connect:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64
```

For information on API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

- Foundational services:

```
export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=1.15.10
export CS_ARCH=amd64
```

For information on IBM Cloud Pak foundational services (common services) CASE versions, see "Table 1. Image versions for offline installation" in Installing IBM Cloud Pak foundational services in an air-gapped environment in the IBM Cloud Pak foundational services documentation.

- EDB:

```
export EDB_CASE_NAME=ibm-cloud-native-postgresql
export EDB_CASE_VERSION=4.18.0
```

b. Connect your portable device to the internet (it does not need to be connected to the network-restricted environment at this time).
c. Download the CASE files to your portable device:
- API Connect:

  ```
  oc ibm-pak get $CASE_NAME --version $CASE_VERSION
  ```

- Foundational services:

  ```
  oc ibm-pak get $CS_CASE_NAME --version $CS_CASE_VERSION
  ```

- EDB:

  ```
  oc ibm-pak get $EDB_CASE_NAME --version $EDB_CASE_VERSION
  ```

4. Mirror the images.
The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before installing API Connect.

a. Generate mirror manifests.
i. Define the environment variable $TARGET_REGISTRY by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace **<target-registry>** with the IP address (or host name) and port of the local registry; for example: **172.16.0.10:5000**. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: **172.16.0.10:5000/registry_ns**.
ii. Generate mirror manifests by running the following commands:
- API Connect:

  ```
  oc ibm-pak generate mirror-manifests $CASE_NAME file://integration --version $CASE_VERSION --final-registry $TARGET_REGISTRY
  ```

- Foundational services:

  ```
  oc ibm-pak generate mirror-manifests $CS_CASE_NAME file://integration --version $CS_CASE_VERSION --final-registry $TARGET_REGISTRY
  ```

- EDB:

  ```
  oc ibm-pak generate mirror-manifests $EDB_CASE_NAME file://integration --version $EDB_CASE_VERSION --final-registry $TARGET_REGISTRY
  ```

If you need to filter for a specific image group, add the parameter **--filter <image_group>** to the command.

The **generate** command creates the following files at ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION, ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION, and ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION:
- catalog-sources.yaml
- catalog-sources-linux-*<arch>*.yaml (if there are architecture-specific catalog sources)
- image-content-source-policy.yaml
- images-mapping-to-filesystem.txt
- images-mapping-from-filesystem.txt

The files are used when mirroring the images to the **TARGET_REGISTRY**.

b. Obtain an entitlement key for the entitled registry where the images are hosted:
i. Log in to the IBM Container Library.
ii. In the Container software library, select Get entitlement key.
iii. In the "Access your container software" section, click Copy key.
iv. Copy the key to a safe location; you will use it to log in to **cp.icr.io** in the next step.

c. Authenticate with the entitled registry where the images are hosted.
The image pull secret allows you to authenticate with the entitled registry and access product images.

i. Run the following command to export the path to the file that will store the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located at $HOME/.docker/config.json on Linux or %USERPROFILE%/.docker/config.json on Windows.

ii. Log in to the **cp.icr.io** registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use **cp** as the username and your entitlement key as the password.

d. Update the API Connect CASE manifest to correctly reference the DataPower Operator image.
Files for the DataPower Operator are now hosted on **icr.io**; however, the CASE manifest still refers to **docker.io** as the image host. To work around this issue, visit Airgap install failure due to 'unable to retrieve source image docker.io' in the DataPower documentation and update the manifest as instructed. The manifest for API Connect (which uses the DataPower Operator) is stored in ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION.

After the manifest is updated, continue to the next step in this procedure.

e. Mirror the images from the internet to the portable device.
i. Define the environment variable $IMAGE_PATH by running the following command:

```
export IMAGE_PATH=<image-path>
```

where **<image-path>** indicates where the files will be stored on the portable device's file system.
ii. Mirror the images to the portable device:
- API Connect:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \
  --filter-by-os '.*' \
  -a $REGISTRY_AUTH_FILE \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --dir "$IMAGE_PATH"
```

- Foundational services:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping-to-filesystem.txt \
  --filter-by-os '.*' \
  -a $REGISTRY_AUTH_FILE \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --dir "$IMAGE_PATH"
```

- EDB:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping-to-filesystem.txt \
  --filter-by-os '.*' \
  -a $REGISTRY_AUTH_FILE \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --dir "$IMAGE_PATH"
```

There might be a slight delay before you see a response to the command.

f. Move the portable device to the restricted-network environment.
The procedure depends on the type of device that you are using:
If you are using a portable computer, disconnect the device from the internet and connect it to the restricted-network environment. The same environment variables can be used.

If you are using portable storage, complete the following steps:

i. Transfer the following files to a device in the restricted-network environment:
   - The **~/.ibm-pak** directory.
   - The contents of the ***<image-path>*** that you specified in the previous step.

ii. Create the same environment variables as on the original device; for example:
   - API Connect:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64
```

   - Foundational services:

```
export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=1.15.10
export CS_ARCH=amd64
```

   - EDB:

```
export EDB_CASE_NAME=ibm-cp-common-services
export EDB_CASE_VERSION=1.15.10
export EDB_ARCH=amd64
```

   - Registry:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

   - Image path:

```
export IMAGE_PATH=<image-path>
```

g. Authenticate with the local registry.
Log in to the local registry using an account that can write images to that registry; for example:

```
podman login $TARGET_REGISTRY
```

If the registry is insecure, add the following flag to the command: **--tls-verify=false**.

h. Mirror the product images to the target registry.

i. If you are using a portable computer, connect it to the restricted-network environment that contains the local registry.
If you are using portable storage, you already transferred files to a device within the restricted-network environment.

ii. Run the following commands to copy the images to the local registry:
   - API Connect:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt \
  -a $REGISTRY_AUTH_FILE \
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --from-dir "$IMAGE_PATH"
```

   - Foundational services:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping-from-filesystem.txt \
  -a $REGISTRY_AUTH_FILE \
```

```
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --from-dir "$IMAGE_PATH"
```

- EDB:

```
oc image mirror \
  -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping-from-filesystem.txt \
  -a $REGISTRY_AUTH_FILE \
  --filter-by-os '.*' \
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --from-dir "$IMAGE_PATH"
```

Note: If the local registry is not secured by TLS, or the certificate presented by the local registry is not trusted by your device, add the **--insecure** option to the command.
There might be a slight delay before you see a response to the command.

i. Configure the target cluster.
Now that images have been mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

i. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

ii. Update the global image pull secret for the cluster as explained in the Red Hat OpenShift Container Platform documentation.
Note: If you have an insecure registry, add the registry to the cluster's **insecureRegistries** list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the TARGET_REGISTRY to **spec.registrySources.insecureRegistries** as shown in the following example:

```
spec:
  registrySources:
    insecureRegistries:
    - insecure0.svc:5001
    - <TARGET_REGISTRY>
```

If the **insecureRegistries** field does not exist, you can add it.

iii. Create the ImageContentSourcePolicy, which instructs the cluster to pull the images from your local registry:

- API Connect:

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

- Foundational services:

```
oc apply -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/image-content-source-policy.yaml
```

- EDB:

```
oc apply -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/image-content-source-policy.yaml
```

iv. Verify that each ImageContentSourcePolicy resource was created:

```
oc get imageContentSourcePolicy
```

v. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before proceeding to the next step.

5. Apply the catalog sources.
Now that you have mirrored images to the target cluster, apply the catalog sources for API Connect and Foundational services. There is no need to apply the catalog source for EDB because the API Connect operator will apply that catalog source in its own namespace when the API Connect operand is created.

In the following steps, replace **<Architecture>** with either **amd64**, **s390x** or **ppc64le** as appropriate for your environment.

a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64

export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=5.0.0
export CS_ARCH=amd64
```

b. Generate the catalog sources and save them in another directory in case you need to replicate this installation in the future.
i. Get the catalog sources:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

cat ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

c. Apply the catalog sources to the cluster.
   i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml
```

```
oc apply -f ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

   ii. Apply any architecture-specific catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

   iii. Confirm that the catalog sources were created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

What to do next

Install API Connect following the steps for your deployment type: Deploying on OpenShift and Cloud Pak for Integration.

Installing a two data center deployment with Cloud Pak for Integration

How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift with Cloud Pak for Integration.

Introduction

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see Two data center warm-standby deployment on Kubernetes and OpenShift.

Familiarize yourself with the Cloud Pak for Integration installation instructions linked from Deploying on OpenShift and Cloud Pak for Integration.
Note: During a 2DCDR installation, the management subsystem remains in a **`Warning`** state until the management subsystems in both data centers synchronize their databases. The management **`Warning`** status prevents the deployment and registration of the portal, gateway, and analytics subsystems. When the management subsystem databases are synchronized and both in **`Ready`** state, the deployment and registration of the other subsystems starts.

Prerequisites

The prerequisites for a 2DCDR deployment with Cloud Pak for Integration are as follows.

- The management and portal endpoints must be the same on both data centers.
- The network latency between data centers must be no more than 80 ms.
- The management cluster in each data center must use the same encryption secret.
- The portal cluster in each data center must use the same encryption secret.
- The ingress-ca certificate must be the same on both data centers.
- The API Connect namespace name must be the same in both clusters.
- The APIConnectCluster CR name must be identical on both clusters.

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a 2DCDR configuration (Two data center warm-standby deployment on Kubernetes and OpenShift).

- **Planning and initial preparation**
  Decide on endpoint, host, and namespace names. Prepare your Cloud Pak for Integration environments for 2DCDR API Connect.
- **Preparing your active data center**
  Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.
- **Preparing your warm-standby data center**
  Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.
- **Installing API Connect on the active data center**
  Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the **`multiSiteHA`** configuration to the API Connect YAML.
- **Installing API Connect on the warm-standby data center**
  Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the **`multiSiteHA`** configuration to the API Connect YAML.

Planning and initial preparation

Decide on endpoint, host, and namespace names. Prepare your Cloud Pak for Integration environments for 2DCDR API Connect.

Procedure

1. Decide on naming. Both data centers must use the same names for:
   - API Connect namespace.
   - API Connect Cluster CR name. Also known as the API Connect instance name, and referred to in this document as **`<apic-instance-name>`**.
   - Management subsystem endpoints
     - **`cloudManagerEndpoint`**
     - **`apiManagerEndpoint`**

    - `platformAPIEndpoint`
    - `consumerAPIEndpoint`
  - Portal subsystem endpoints
    - `portalAdminEndpoint`
    - `portalUIEndpoint`

  Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see Backup and restore requirements for a 2DCDR deployment.
2. Create a namespace for your API Connect deployment in both data centers.
   The API Connect namespace must be the same in both data centers, set it with:

   ```
   oc create ns <namespace>
   ```

3. Install the API Connect operators in both your data centers. See Installing operators. Return here when the operators are installed.
4. Decide which data center will start as the active and which will start as the warm-standby.
5. On the CLI of the active data center, where you run `oc` commands, create a directory called `2dcdr-active-yamls`. On the CLI of the warm-standby data center, create a directory called `2dcdr-ws-yamls`.

What to do next

Proceed to prepare the replication certificates, secrets, and issuers on your active data center Preparing your active data center.

Preparing your active data center

Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.

About this task

All operations are done on the CLI, in the `2dcdr-active-yamls` directory you created in Planning and initial preparation.

In the yaml files and commands that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR, and `<namespace>` with your API Connect namespace name (which will be the same for both Management and Portal subsystems). As decided in Planning and initial preparation.

Procedure

1. Create the encryption key secrets for the Management and Portal subsystems.
   a. Run the following command to create a file that contains a random string, which is used to create the management encryption key secret:

   ```
   cat /dev/urandom | head -c63 | base64 -w0 > mgmt-enc-key.txt
   ```

   b. Run the following command to create the management encryption key secret:

   ```
   oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management
   namespace>
   ```

   c. Confirm that the secret was created successfully by running:

   ```
   oc get secrets -n <management namespace> | grep mgmt-encryption-key

   mgmt-encryption-key         Opaque                                1      83s
   ```

   d. Run the following command to create a file that contains a random string, which is used to create the portal encryption key secret:

   ```
   cat /dev/urandom | head -c63 | base64 -w0 > ptl-enc-key.txt
   ```

   e. Run the following command to create the portal encryption key secret:

   ```
   oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
   ```

   f. Confirm that the secret was created successfully by running:

   ```
   oc get secrets -n <portal namespace> | grep ptl-encryption-key

   ptl-encryption-key          Opaque                                1      15s
   ```

2. Create a self-signed issuer.
   a. Create a file that is called `issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

   ```
   apiVersion: cert-manager.io/v1
   kind: Issuer
   metadata:
     name: <apic-instance-name>-self-signed
   spec:
     selfSigned: {}
   ```

   b. Apply this yaml file with:

   ```
   oc apply -f issuer.yaml -n <namespace>
   ```

   c. Verify that the issuer was created with:

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

```
NAME                READY   AGE
apic-self-signed    True    25s
```

3. Create an ingress-ca certificate:
   a. Create a file that is called `self-signed-issuer-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ingress-ca
spec:
  commonName: ingress-ca
  duration: 87600h0m0s
  isCA: true
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-self-signed
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ingress-ca
```

   b. Apply this yaml file with:

```
oc apply -f self-signed-issuer-cert.yaml -n <namespace>
```

   c. Verify that the certificate was created with:

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

```
NAME                          READY   SECRET                        AGE     EXPIRATION
apic-ingress-ca               True    apic-ingress-ca               11s     2032-08-15T13:01:47Z
```

4. Create the ingress issuer.
   a. Create a file `ingress-issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: <apic-instance-name>-ingress-issuer
spec:
  ca:
    secretName: <apic-instance-name>-ingress-ca
```

   b. Apply this yaml file with:

```
oc apply -f ingress-issuer.yaml -n <namespace>
```

   c. Verify that the issuer was created with:

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

```
NAME                  READY   AGE
apic-ingress-issuer   True    20s
```

5. Create the TLS client replication certificates for Management and Portal.
   a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-mgmt-replication-client
spec:
  commonName: <apic-instance-name>-mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-mgmt-replication-client
```

   b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

   c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>
```

```
NAME                                          READY   SECRET                                        AGE
EXPIRATION
...
<apic-instance-name>-mgmt-replication-client   True    <apic-instance-name>-mgmt-replication-client   16m     2024-
08-17T13:04:27Z
```

   d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ptl-replication-client
spec:
  commonName: <apic-instance-name>-ptl-replication-client
```

```
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

```
NAME                                              READY   SECRET                                            AGE
EXPIRATION
...
<apic-instance-name>-ptl-replication-client   True    <apic-instance-name>-ptl-replication-client   16m     2024-08-
17T13:04:27Z
```

6. Export the ingress-ca issuer secret.
   The ingress-ca issuer secret must be the same on the warm-standby data center. Export it from the active so that it can be imported on the warm-standby data center.
   a. Run the following command to export the secret to a file called `ca-issuer-secret.yaml`:

```
oc get secret <apic-instance-name>-ingress-ca -o yaml -n <namespace>  > ca-issuer-secret.yaml
```

   b. Edit the `ca-issuer-secret.yaml` file to remove the creationTimestamp, resourceVersion, uid, and managedFields. Remove the labels and annotations sections completely. The contents should look like this:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: <apic-instance-name>-ingress-ca
  namespace: <namespace>
type: kubernetes.io/tls
```

7. Copy the following files from your `2dcdr-active-yamls` directory to the `2dcdr-ws-yamls` directory in your warm-standby data center.

```
ca-issuer-secret.yaml
mgmt-enc-key.txt
ptl-enc-key.txt
```

   These files are required to ensure that the ingress-ca and encryption secrets on both sites are the same.
   Tip: To save time during the preparation of your warm-standby data center you can also copy these yaml files:

```
ptl-tls-client-cert.yaml
mgmt-tls-client-cert.yaml
ingress-issuer.yaml
```

What to do next

Create your secrets, certs, and issuers on your warm-standby data center: Preparing your warm-standby data center.

Preparing your warm-standby data center

Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.

About this task

All operations are done on the CLI, in the **`2dcdr-ws-yamls`** directory you created in Planning and initial preparation.

In the YAML files and commands that are shown here, replace **`<apic-instance-name>`** with the name you intend to use for your API Connect Cluster CR, and **`<namespace>`** with your API Connect namespace name (which will be the same for both Management and Portal subsystems). As decided in Planning and initial preparation.

Procedure

1. Import the ingress-ca issuer secret.
   This secret was created on your active data center here: Export the ingress-ca issuer secret
   a. Copy the `ca-issuer-secret.yaml` file to your warm-standby data center (if not done already).
   b. Create the secret with:

```
oc apply -f ca-issuer-secret.yaml -n <namespace>
```

   c. Verify that the secret was created with:

```
oc get secret -n <namespace>

NAME                                                TYPE                                DATA   AGE
<apic-instance-name>-ingress-ca                     kubernetes.io/tls                   3      19h
```

2. Create the ingress issuer.
   a. Create a file `ingress-issuer.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: <apic-instance-name>-ingress-issuer
spec:
  ca:
    secretName: <apic-instance-name>-ingress-ca
```

   b. Apply this yaml file with:

```
oc apply -f ingress-issuer.yaml -n <namespace>
```

   c. Verify that the issuer was created with:

```
oc get issuer -n <namespace>

NAME                  READY   AGE
apic-ingress-issuer   True    20s
```

3. Create the encryption key secrets for the Management and Portal subsystems, by using the random string generated on the active data center.
   Use the `mgmt-enc-key.txt` and `ptl-enc-key.txt` files that were copied to your warm-standby data center here: Copy keys to warm-standby.
   a. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management
namespace>
```

   b. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key

mgmt-encryption-key        Opaque                                1      83s
```

   c. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

   d. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key

ptl-encryption-key         Opaque                                1      15s
```

4. Create the TLS client replication certificates for Management and Portal.
   a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-mgmt-replication-client
spec:
  commonName: <apic-instance-name>-mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-mgmt-replication-client
```

   b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

   c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>

NAME                                          READY   SECRET                                        AGE
EXPIRATION
...
<apic-instance-name>-mgmt-replication-client   True    <apic-instance-name>-mgmt-replication-client   16m     2024-
08-17T13:04:27Z
```

   d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents, replacing `<apic-instance-name>`:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <apic-instance-name>-ptl-replication-client
spec:
  commonName: <apic-instance-name>-ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: <apic-instance-name>-ingress-issuer
  renewBefore: 720h0m0s
```

```
  privateKey:
    rotationPolicy: Always
  secretName: <apic-instance-name>-ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>
```

```
NAME                                          READY   SECRET                                        AGE
EXPIRATION
...
<apic-instance-name>-ptl-replication-client   True    <apic-instance-name>-ptl-replication-client   16m     2024-08-
17T13:04:27Z
```

What to do next

Install API Connect in your active data center: Installing API Connect on the active data center.

Installing API Connect on the active data center

Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the **multiSiteHA** configuration to the API Connect YAML.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-ca               kubernetes.io/tls                     3      3d21h
<apic-instance-name>-mgmt-replication-client  kubernetes.io/tls                     3      3d21h
<apic-instance-name>-ptl-replication-client   kubernetes.io/tls                     3      3d21h

oc get certs -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-ca               True    <apic-instance-name>-ingress-ca               3d21h   2032-08-
15T13:01:47Z
<apic-instance-name>-mgmt-replication-client  True    <apic-instance-name>-mgmt-replication-client  3d21h   2024-08-
17T13:04:27Z
<apic-instance-name>-ptl-replication-client   True    <apic-instance-name>-ptl-replication-client   3d21h   2024-08-
17T13:04:26Z

oc get issuer -n <namespace> | grep <apic-instance-name>
<apic-instance-name>-ingress-issuer   True    3d21h
<apic-instance-name>-self-signed      True    3d21h
```

Where **<apic-instance-name>** is the name you intend to use for your API Connect cluster CR, and **<namespace>** is the namespace you created for API Connect.

About this task

In the yaml files that are shown here, replace **<apic-instance-name>** with the name you intend to use for your API Connect Cluster CR. As decided in Planning and initial preparation. Set **<active data center ingress domain>** and **<warm-standby data center ingress domain>** to their appropriate values, which can be determined by running this command in each data center:

```
oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

Procedure

1. Follow the API Connect installation steps using the Platform UI, as described in the Cloud Pak for Integration documentation https://www.ibm.com/docs/en/cloud-paks/cp-integration. At the point where you specify the configuration properties of your API Connect instance, switch to the YAML tab so you can edit the YAML directly.
   Remember: Use the same **<apic-instance-name>** value when you specify the API Connect instance name in the Cloud Pak for Integration Platform UI.
2. Add the management **multiSiteHA** section to the YAML file under the **spec:** section.

```
  management:
    encryptionSecret:
      secretName: mgmt-encryption-key
    multiSiteHA:
      mode: active
      replicationEndpoint:
        annotations:
          cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
        hosts:
        - name: mgmt-replication.<active data center ingress domain>
          secretName: <apic-instance-name>-mgmt-replication-server
      replicationPeerFQDN: mgmt-replication.<warm-standby data center ingress domain>
      tlsClient:
        secretName: <apic-instance-name>-mgmt-replication-client
```

3. Add the portal **multiSiteHA** section to the YAML file under the **spec:** section:

```
  portal:
    portalAdminEndpoint:
```

```
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
    - name: <external load balanced portal admin hostname>
      secretName: portal-admin
  portalUIEndpoint:
    annotations:
      cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
    hosts:
    - name: <external load balanced portal web hostname>
      secretName: portal-web
  encryptionSecret:
    secretName: ptl-encryption-key
  multiSiteHA:
    mode: active
    replicationEndpoint:
      annotations:
        cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
      hosts:
      - name: ptl-replication.<active data center ingress domain>
        secretName: <apic-instance-name>-ptl-replication-server
    replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
    tlsClient:
      secretName: <apic-instance-name>-ptl-replication-client
```

4. Continue the installation steps as described in Deploying on OpenShift and Cloud Pak for Integration.

Results

Confirm that the management subsystem is ready, but in `Warning` state with `oc get mgmt`:

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

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in CR
for details   8m59s
```

```
oc get mgmt -n <namespace> -o yaml
```

```
...
status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working, install or upgrade in progress.",
      "reason": "na",
      "status": "True",
      "type": "Pending"
    }
```

The management CR is expected to report the status of `Warning` until the warm-standby management subsystem is deployed, and both management subsystems complete data replication. When you see the status message "Management is ready. HA Status Warning - see HAStatus in CR for details", you can move on to Installing API Connect on the warm-standby data center.
Note: The portal, analytics, and gateway subsystems are not deployed until the management replication is complete and the management CRs in both data centers report the following:

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

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Running   10.0.7.0-0   10.0.7.0-0           Management is ready. HA status Ready - see HAStatus in CR for
details               8m59s
```

```
oc get mgmt -n <namespace> -o yaml
```

```
...
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

What to do next

Install API Connect on your warm-standby data center: Installing API Connect on the warm-standby data center.

Installing API Connect on the warm-standby data center

Follow the installation steps for API Connect described in the Cloud Pak for Integration documentation, adding the `multiSiteHA` configuration to the API Connect YAML.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get certificate -n <namespace> | grep <apic-instance-name>
NAME                              READY   SECRET                            AGE     EXPIRATION
apic-mgmt-replication-client      True    apic-mgmt-replication-client      57s     2024-08-24T09:45:03Z
apic-ptl-replication-client       True    apic-ptl-replication-client       45s     2024-08-24T09:45:17Z

[root@api.iaincp4ilarge2.cp.fyre.ibm.com ~]# oc get secrets  -n <namespace> | grep <apic-instance-name>
NAME                                TYPE                                  DATA   AGE
apic-ingress-ca                     kubernetes.io/tls                     3      4m6s
apic-mgmt-replication-client        kubernetes.io/tls                     3      2m40s
apic-ptl-replication-client         kubernetes.io/tls                     3      2m26s

[root@api.iaincp4ilarge2.cp.fyre.ibm.com ~]# oc get issuer -n <namespace> | grep <apic-instance-name>
NAME                  READY   AGE
apic-ingress-issuer   True    4m8s
```

Where `<apic-instance-name>` is the name you intend to use for your API Connect cluster CR, and `<namespace>` is the namespace you created for API Connect.

About this task

In the yaml files that are shown here, replace `<apic-instance-name>` with the name you intend to use for your API Connect Cluster CR. As decided in Planning and initial preparation. Set `<active data center ingress domain>` and `<warm-standby data center ingress domain>` to their appropriate values, which can be determined by running this command in each data center:

```
oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

Procedure

1. Follow the API Connect installation steps using the Platform UI, as described in the Cloud Pak for Integration documentation https://www.ibm.com/docs/en/cloud-paks/cp-integration. At the point where you specify the configuration properties of your API Connect instance, switch to the YAML tab so you can edit the YAML directly.
   Remember: Use the same `<apic-instance-name>` value when you specify the API Connect instance name in the Cloud Pak for Integration Platform UI.
2. Add the management `multiSiteHA` section to the YAML file under the `spec:` section.

   ```
   management:
     encryptionSecret:
       secretName: mgmt-encryption-key
     multiSiteHA:
       mode: passive
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
         hosts:
         - name: mgmt-replication.<warm-standby data center ingress domain>
           secretName: <apic-instance-name>-mgmt-replication-server
       replicationPeerFQDN: mgmt-replication.<active data center ingress domain>
       tlsClient:
         secretName: <apic-instance-name>-mgmt-replication-client
   ```

   Note: Warm-standby is referred to as 'passive' in the CR YAML.
3. Add the portal `multiSiteHA` section to the YAML file under the `spec:` section:

   ```
   portal:
     portalAdminEndpoint:
       annotations:
         cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
       hosts:
       - name: <external load balanced portal admin hostname>
         secretName: portal-admin
     portalUIEndpoint:
       annotations:
         cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
       hosts:
       - name: <external portal load balanced web hostname>
         secretName: portal-web
     encryptionSecret:
       secretName: ptl-encryption-key
     multiSiteHA:
       mode: passive
       replicationEndpoint:
         annotations:
           cert-manager.io/issuer: <apic-instance-name>-ingress-issuer
         hosts:
         - name: ptl-replication.<warm-standby data center ingress domain>
           secretName: <apic-instance-name>-ptl-replication-server
       replicationPeerFQDN: ptl-replication.<active data center ingress domain>
       tlsClient:
         secretName: <apic-instance-name>-ptl-replication-client
   ```

4. Disable the configurator service, as this will have already run on the active site.
   a. Add the following section to the API Connect cluster CR:

      ```
      disabledServices:
        - configurator
      ```

5. Continue the installation steps as described in Deploying on OpenShift and Cloud Pak for Integration.

Results

Installation can take around 1 hour to complete.

While the management subsystems on the warm-standby and active data centers are synchronizing, the management status reports **Warning**, and the **haStatus** reports **pending**:

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

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in CR
for details   8m59s

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

When the management database replication between sites is complete, the management status reports **Running**, and **status.haStatus** reports **Ready**:

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Running   10.0.7.0-0   10.0.7.0-0           Management is ready. HA status Ready - see HAStatus in CR for
details               8m59s

status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

After the management subsystems in the active and warm-standby sites are synchronized, the other subsystems are deployed. Verify on both active and warm-standby sites that all subsystems reach **Running** state:

```
oc get all -n <namespace>
...
NAME                                                          READY   STATUS    VERSION    RECONCILED VERSION   AGE
analyticscluster.analytics.apiconnect.ibm.com/apis-minimum-a7s   n/n     Running   10.0.7.0   10.0.7.0-1281        27m

NAME                                                  READY   STATUS   VERSION    RECONCILED VERSION   AGE
apiconnectcluster.apiconnect.ibm.com/apis-minimum     n/n     Ready    10.0.7.0   10.0.7.0-1281        2d20h

NAME                                                  PHASE     READY   SUMMARY                          VERSION    AGE
datapowerservice.datapower.ibm.com/apis-minimum-gw    Running   True    StatefulSet replicas ready: 1/1  10.5.0.0   25m

NAME                                                  PHASE     LAST EVENT   WORK PENDING   WORK IN-PROGRESS   AGE
datapowermonitor.datapower.ibm.com/apis-minimum-gw    Running                false          false              25m

NAME                                                 READY   STATUS    VERSION    RECONCILED VERSION   AGE
gatewaycluster.gateway.apiconnect.ibm.com/apis-minimum-gw   n/n     Running   10.0.7.0   10.0.7.0-1281        27m

NAME                                                        READY   STATUS    VERSION    RECONCILED VERSION   AGE
managementcluster.management.apiconnect.ibm.com/apis-minimum-mgmt   8/8     Running   10.0.7.0   10.0.7.0-1281        2d20h

NAME                                                 READY   STATUS    VERSION    RECONCILED VERSION   AGE
portalcluster.portal.apiconnect.ibm.com/apis-minimum-ptl   n/n     Running   10.0.7.0   10.0.7.0-1281        27m
```

Note: The warm-standby site has fewer **managementcluster** pods than the active site.
If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the **haStatus** output shows the reason, for example::

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in CR
for details   8m59s

status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working",
      "reason": "na",
      "status": "True",
      "type": "PeerNotReachable"
    }
```

If the warning persists, see Troubleshooting a two data center deployment.

You can validate that your portal deployments are synchronizing by running **oc get pods** on both the active and warm-standby data centers. Confirm that the number and names of the pods all match (the UUIDs in the names might be different on each site), and that all are in the **Ready** state.

For additional replication verification checks, see Verifying replication between data centers. It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See Failing over to the warm-standby.

What to do next

Configure your deployment: Cloud Manager configuration checklist.
Important: It is strongly recommended to take complete backups of both sites, for disaster recovery in case the API Connect installation in both data centers is lost, or if a replication failure occurs. Complete backups consist of your infrastructure configuration backup and your subsystem database backups. See Backup and restore requirements for a 2DCDR deployment.

Installing a two data center deployment on OpenShift

How to deploy a two data center disaster recovery (2DCDR) deployment on OpenShift.

Introduction

Ensure that you understand the concepts of 2DCDR in API Connect. For more information, see Two data center warm-standby deployment on Kubernetes and OpenShift.

Familiarize yourself with the OpenShift installation instructions linked from Deploying on OpenShift and Cloud Pak for Integration.

Note: 2DCDR is not supported on OpenShift where the top-level CR is used. For 2DCDR on OpenShift, you must install API Connect as individual subsystems. See Installing API Connect in a single namespace on OpenShift and Installing API Connect subsystems in different namespaces or environments on OpenShift.

Prerequisites

The prerequisites for a 2DCDR deployment on OpenShift are as follows.

- The management and portal endpoints must be the same on both data centers.
- The network latency between data centers must be no more than 80 ms.
- The management cluster in each data center must use the same encryption secret.
- The portal cluster in each data center must use the same encryption secret.
- The ingress-ca certificate must be the same on both data centers.

Restrictions:

- API Connect is not supported on a FIPS-enabled environment.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a 2DCDR configuration (Two data center warm-standby deployment on Kubernetes and OpenShift).

- **Planning and initial preparation**
  Decide on endpoint names. Prepare your OpenShift environments for 2DCDR API Connect.
- **Preparing your active data center**
  Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.
- **Preparing your warm-standby data center**
  Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.
- **Installing API Connect on the active data center**
  Add the `multiSiteHA` configuration to the API Connect Management and Portal CR YAML files before they are applied.
- **Installing API Connect on the warm-standby data center**
  Add the `multiSiteHA` configuration to the API Connect Management and Portal YAML files before they are applied.

Planning and initial preparation

Decide on endpoint names. Prepare your OpenShift environments for 2DCDR API Connect.

Procedure

1. Decide on naming. Both data centers must use the same names for:
   - Management subsystem endpoints
     - `cloudManagerEndpoint`
     - `apiManagerEndpoint`
     - `platformAPIEndpoint`
     - `consumerAPIEndpoint`
   - Portal subsystem endpoints
     - `portalAdminEndpoint`
     - `portalUIEndpoint`

   Each data center must have a different `siteName`. For example, `dallas` in data center 1, and `raleigh` in data center 2. The site name can contain only `a-z` and `0-9` characters. You set `siteName` in the `ManagementCluster` and `PortalCluster` CRs when the subsystems are first deployed. You cannot change the name after deployment. If you don't specify `siteName` at first deployment, a random `siteName` is created for you.

   Important: Each data center must use a different backup path for your Management backups. For portal backups each data center must use the same backup path. For more information, see Backup and restore requirements for a 2DCDR deployment.
2. Install the API Connect operators in both your data centers. See Installing operators for a single namespace API Connect installation or Installing operators for a multi namespace API Connect installation, depending on whether all your API Connect subsystems are to be in the same namespace or not. Return here when the operators are installed.
3. If not already done, create namespaces for your API Connect subsystems in both data centers.

   ```
   oc create ns <namespace>
   ```

the data centers do not have to use the same namespace name.
4. Decide which data center will start as the active and which will start as the warm-standby.
5. On the CLI of the active data center, where you run `oc` commands, create a directory called `2dcdr-active-yamls`. On the CLI of the warm-standby data center, create a directory called `2dcdr-ws-yamls`.

What to do next

Proceed to prepare the replication certificates, secrets, and issuers on your active data center Preparing your active data center.

Preparing your active data center

Create the secrets, certificates, and issuers that are needed for your active data center to replicate with the warm-standby.

About this task

All operations are done on the CLI, in the `2dcdr-active-yamls` directory you created in Planning and initial preparation.

In the yaml files and commands that are shown here, replace `<namespace>` with the name of the corresponding subsystem namespace.

Procedure

1. Create the encryption key secrets for the Management and Portal subsystems.
   a. Run the following command to create a file that contains a random string, which is used to create the management encryption key secret:

   ```
   cat /dev/urandom | head -c63 | base64 -w0 > mgmt-enc-key.txt
   ```

   b. Run the following command to create the management encryption key secret:

   ```
   oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management
   namespace>
   ```

   c. Confirm that the secret was created successfully by running:

   ```
   oc get secrets -n <management namespace> | grep mgmt-encryption-key

   mgmt-encryption-key         Opaque                                1      83s
   ```

   d. Run the following command to create a file that contains a random string, which is used to create the portal encryption key secret:

   ```
   cat /dev/urandom | head -c63 | base64 -w0 > ptl-enc-key.txt
   ```

   e. Run the following command to create the portal encryption key secret:

   ```
   oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
   ```

   f. Confirm that the secret was created successfully by running:

   ```
   oc get secrets -n <portal namespace> | grep ptl-encryption-key

   ptl-encryption-key          Opaque                                1      15s
   ```

2. If all your API Connect subsystems are to be installed in the same namespace, then create your certificates, issuers, and secrets as described here: Setting up a certificate issuer. Return here when complete, do not proceed to install the subsystems.
3. If your API Connect subsystems are to be installed in different namespaces, then create your certificates, issuers, and secrets for your Management subsystem as described here Create Management subsystem issuers and secrets. For your Portal subsystem, follow the steps here Create Portal subsystem issuers and secretes. Return here when complete, do not proceed to install either subsystem.
4. Create the TLS client replication certificates for Management and Portal.
   a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents:

   ```
   apiVersion: cert-manager.io/v1
   kind: Certificate
   metadata:
     name: mgmt-replication-client
   spec:
     commonName: mgmt-replication-client
     duration: 17520h0m0s
     issuerRef:
       kind: Issuer
       name: ingress-issuer
     renewBefore: 720h0m0s
     privateKey:
       rotationPolicy: Always
     secretName: mgmt-replication-client
   ```

   b. Apply this yaml file with:

   ```
   oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
   ```

   c. Verify that the certificate was created with:

   ```
   oc get certs -n <management namespace>

   NAME                        READY   SECRET                      AGE     EXPIRATION
   ...
   mgmt-replication-client     True    mgmt-replication-client     16m     2024-08-17T13:04:27Z
   ```

d. Create a file **ptl-tls-client-cert.yaml** and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ptl-replication-client
spec:
  commonName: ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>

NAME                        READY   SECRET                      AGE     EXPIRATION
...
ptl-replication-client      True    ptl-replication-client      16m     2024-08-17T13:04:27Z
```

5. Export the ingress-ca issuer secret.
   The ingress-ca issuer secret must be the same on the warm-standby data center. Export it from the active so that it can be imported on the warm-standby data center.
   Note: If you are deploying API Connect subsystems in different namespaces, then you export the Management subsystem ingress-ca secret.
   a. Run the following command to export the secret to a file called **ca-issuer-secret.yaml**:

```
oc get secret ingress-ca -o yaml -n <management namespace>  > ca-issuer-secret.yaml
```

   b. Edit the **ca-issuer-secret.yaml** file to remove the creationTimestamp, resourceVersion, uid, namespace, and managedFields. Remove the labels and annotations sections completely. The resulting contents should look like this:

```
apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
type: kubernetes.io/tls
```

6. Copy the following files from your **2dcdr-active-yamls** directory to the **2dcdr-ws-yamls** directory in your warm-standby data center.

```
ca-issuer-secret.yaml
mgmt-enc-key.txt
ptl-enc-key.txt
```

   These files are required to ensure that the ingress-ca and encryption secrets on both sites are the same.
   Tip: To save time during the preparation of your warm-standby data center you can also copy these yaml files:

```
ptl-tls-client-cert.yaml
mgmt-tls-client-cert.yaml
```

7. If you are installing the Portal subsystem in a different namespace to the Management subsystem on the active data center, use the **ca-issuer-secret.yaml** file to create the issuer in your portal namespace. Follow the 'Before you begin steps here: Create issuer. Do not proceed to install the Portal subsystem after creating the issuer.

What to do next

Prepare your warm-standby data center Preparing your warm-standby data center.

Preparing your warm-standby data center

Create the secrets, certificates, and issuers that are needed for your warm-standby data center to replicate with the active.

About this task

All operations are done on the CLI, in the **2dcdr-ws-yamls** directory you created in Planning and initial preparation.

In the yaml files and commands that are shown here, replace **<namespace>** with the name of the corresponding subsystem namespace.

Procedure

1. Import the ingress-ca issuer secret.
   This secret was created on your active data center here: Preparing your active data center
   a. If you did not do it as part of Preparing your active data center, copy the **ca-issuer-secret.yaml** file to your warm-standby data center.

b. Create the secret with:

```
oc apply -f ca-issuer-secret.yaml -n <namespace>
```

c. Verify that the secret was created with:

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

```
NAME                              TYPE                                  DATA   AGE
ingress-ca                        kubernetes.io/tls                     3      19h
```

d. If your Portal subsystem is in a different namespace in your warm-standby data center, then repeat the `oc apply` operation against the portal namespace.

2. Create a file that is called `warm-standby-certs.yaml` and paste in:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: portal-admin-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "portal-admin-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: portal-admin-client
  secretName: portal-admin-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "portal-admin-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-client-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-client-client"
  }
```

```
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-client-client
  secretName: gateway-client-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-client-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: analytics-ingestion-client
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "analytics-ingestion-client"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: analytics-ingestion-client
  secretName: analytics-ingestion-client
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "management"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "analytics-ingestion-client"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-service
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-service"
  }
spec:
  subject:
    organizations:
    - cert-manager
  commonName: gateway-service
  secretName: gateway-service
  issuerRef:
    name: ingress-issuer
  usages:
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "gatewaycluster"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-service"
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: gateway-peering
  labels: {
    app.kubernetes.io/instance: "gatewaycluster",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "gateway-peering"
  }
spec:
  subject:
    organizations:
```

```
    - cert-manager
  commonName: gateway-peering
  secretName: gateway-peering
  issuerRef:
    name: ingress-issuer
  usages:
  - "server auth"
  - "client auth"
  - "signing"
  - "key encipherment"
  duration: 17520h # 2 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  secretTemplate:
    labels:
      app.kubernetes.io/instance: "gatewaycluster"
      app.kubernetes.io/managed-by: "ibm-apiconnect"
      app.kubernetes.io/name: "gateway-peering"
```

3. Apply this file by running against your Management subsystem namespace:

```
oc apply -f warm-standby-certs.yaml -n <namespace>
```

4. Verify that the certificates, secrets, and issuers were created with:

```
oc get secrets -n <namespace>
NAME                          TYPE                                  DATA   AGE
analytics-ingestion-client    kubernetes.io/tls                     3      12s
gateway-client-client         kubernetes.io/tls                     3      21s
gateway-peering               kubernetes.io/tls                     3      11s
gateway-service               kubernetes.io/tls                     3      21s
ingress-ca                    kubernetes.io/tls                     3      18m
portal-admin-client           kubernetes.io/tls                     3      17s

oc get certificates -n <namespace>
NAME                          READY   SECRET                        AGE     EXPIRATION
analytics-ingestion-client    True    analytics-ingestion-client    2m18s   2024-09-04T12:27:38Z
gateway-client-client         True    gateway-client-client         2m19s   2024-09-04T12:27:28Z
gateway-peering               True    gateway-peering               2m17s   2024-09-04T12:27:39Z
gateway-service               True    gateway-service               2m18s   2024-09-04T12:27:28Z
portal-admin-client           True    portal-admin-client           2m19s   2024-09-04T12:27:33Z

oc get issuers -n <namespace>
NAME                 READY   AGE
ingress-issuer       True    18m
selfsigning-issuer   True    2m42s
```

5. If your Portal subsystem is in a different namespace in your warm-standby data center, then to create your certificates, issuers, and secrets in the Portal subsystem namespace:
   a. Create a file that is called `warm-standby-ptl-certs.yaml` and paste in:

```
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

---

apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigning-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "selfsigning-issuer"
  }
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ingress-issuer
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-issuer"
  }
spec:
  ca:
    secretName: ingress-ca
```

   b. Apply this file to your Portal subsystem namespace:

```
oc apply -f warm-standby-certs.yaml -n <namespace>
```

c. Verify that your issuers and secrets were created successfully:

```
oc get issuers -n <portal namespace>
NAME                READY   AGE
ingress-issuer      True    43s
selfsigning-issuer  True    43s

oc get secrets -n <portal namespace>
NAME                        TYPE                                DATA   AGE
ingress-ca                  kubernetes.io/tls                   3      4m18s
```

6. Create the encryption key secrets for the Management and Portal subsystems, by using the random string generated on the active data center.
Use the `mgmt-enc-key.txt` and `ptl-enc-key.txt` files that were copied to your warm-standby data center when you followed Preparing your active data center.

a. Run the following command to create the management encryption key secret:

```
oc create secret generic mgmt-encryption-key --from-file=encryption_secret.bin=mgmt-enc-key.txt -n <management
namespace>
```

b. Confirm that the secret was created successfully by running:

```
oc get secrets -n <management namespace> | grep mgmt-encryption-key

mgmt-encryption-key         Opaque                              1      83s
```

c. Run the following command to create the portal encryption key secret:

```
oc create secret generic ptl-encryption-key --from-file=encryption_secret=ptl-enc-key.txt -n <portal namespace>
```

d. Confirm that the secret was created successfully by running:

```
oc get secrets -n <portal namespace> | grep ptl-encryption-key

ptl-encryption-key          Opaque                              1      15s
```

7. Create the TLS client replication certificates for Management and Portal.

a. Create a yaml file that is called `mgmt-tls-client-cert.yaml` and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mgmt-replication-client
spec:
  commonName: mgmt-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: mgmt-replication-client
```

b. Apply this yaml file with:

```
oc apply -f mgmt-tls-client-cert.yaml -n <management namespace>
```

c. Verify that the certificate was created with:

```
oc get certs -n <management namespace>

NAME                      READY   SECRET                    AGE     EXPIRATION
...
mgmt-replication-client   True    mgmt-replication-client   16m     2024-08-17T13:04:27Z
```

d. Create a file `ptl-tls-client-cert.yaml` and paste in the following contents:

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ptl-replication-client
spec:
  commonName: ptl-replication-client
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: ptl-replication-client
```

e. Apply this yaml file with:

```
oc apply -f ptl-tls-client-cert.yaml -n <portal namespace>
```

f. Verify that the certificate was created with:

```
oc get certs -n <portal namespace>

NAME                      READY   SECRET                    AGE     EXPIRATION
...
ptl-replication-client    True    ptl-replication-client    16m     2024-08-17T13:04:27Z
```

What to do next

Install API Connect on your active data center Installing API Connect on the active data center.

Installing API Connect on the active data center

Add the **multiSiteHA** configuration to the API Connect Management and Portal CR YAML files before they are applied.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace>
ingress-ca                kubernetes.io/tls                     3      3d21h
mgmt-replication-client   kubernetes.io/tls                     3      3d21h
ptl-replication-client    kubernetes.io/tls                     3      3d21h

oc get certs -n <namespace>
ingress-ca                True    ingress-ca                 3d21h   2032-08-15T13:01:47Z
mgmt-replication-client   True    mgmt-replication-client    3d21h   2024-08-17T13:04:27Z
ptl-replication-client    True    ptl-replication-client     3d21h   2024-08-17T13:04:26Z

oc get issuer -n <namespace>
ingress-issuer   True    3d21h
self-signed      True    3d21h
```

Where **<namespace>** are the namespaces you created for API Connect.
Note: If you have separate namespaces for Management and Portal subsystems, then run the check in both namespaces. You should not see the **mgmt-replication-client** in the portal namespace, nor the **ptl-replication-client** in the management namespace.

Procedure

1. Follow these steps for installing the Management subsystem on OpenShift Installing the Management subsystem in a shared namespace, but add the **encryptionSecret**
   and **multiSiteHA** section to the **mgmt_cr.yaml** file:

   ```
   siteName: <dc1 site name>
   encryptionSecret:
     secretName: mgmt-encryption-key
   multiSiteHA:
     mode: active
     replicationEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: mgmt-replication.<active data center ingress domain>
         secretName: mgmt-replication-server
     replicationPeerFQDN: mgmt-replication.<warm-standby data center ingress domain>
     tlsClient:
       secretName: mgmt-replication-client
   ```

   where
   - **<dc1 site name>** is the site name for this data center, as decided in Planning and initial preparation.
   - **<active data center ingress domain>** and **<warm-standby data center ingress domain>** can be determined by running the command **oc get ingresses.config/cluster -o jsonpath={.spec.domain}** in each data center.

   Note: If you are installing the Management subsystem in a separate namespace, these same steps apply.
2. Apply the update **mgmt_cr.yaml** file as directed in Installing the Management subsystem in a shared namespace. You can monitor the deployment with:

   ```
   oc get ManagementCluster -n <management namespace>
   NAME         READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE AGE
   management   n/n   Running   10.0.7.0   10.0.7.0-1281        Management is ready. HA status Warning - see HAStatus in CR
   for details 74m
   ```

3. Follow these steps for installing the Portal subsystem on OpenShift Installing the Portal subsystem in a shared namespace, but add the **encryptionSecret** and **multiSiteHA** section to the **portal_cr.yaml** file.

   ```
   siteName: <dc1 site name>
   encryptionSecret:
     secretName: ptl-encryption-key
   multiSiteHA:
     mode: active
     replicationEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: ptl-replication.<active data center ingress domain>
         secretName: ptl-replication-server
     replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
     tlsClient:
       secretName: ptl-replication-client
   ```

   where
   - **<dc1 site name>** is the site name for this data center, as decided in Planning and initial preparation.

- **<active data center ingress domain>** and **<warm-standby data center ingress domain>** can be determined by running the command **oc get ingresses.config/cluster -o jsonpath={.spec.domain}** in each data center.

Note: If you are installing the Portal subsystem in a separate namespace, these same steps apply.

4. Apply the updated **portal_cr.yaml** file as directed in Installing the Portal subsystem in a shared namespace.

```
oc get PortalCluster -n <portal namespace>
NAME      READY    STATUS    VERSION    RECONCILED VERSION    AGE
portal    6/6      Running   10.0.7.0   10.0.7.0-1281         49m
```

Results

Confirm that the management subsystem is ready, but in **Warning** state with **oc get mgmt**:

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

```
NAME          READY    STATUS    VERSION      RECONCILED VERSION    MESSAGE
AGE
management    n/n      Warning   10.0.7.0-0   10.0.7.0-0            Management is ready. HA Status Warning - see HAStatus in CR
for details    8m59s
```

```
oc get mgmt -n <namespace> -o yaml
```

```
...
status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working, install or upgrade in progress.",
      "reason": "na",
      "status": "True",
      "type": "Pending"
    }
```

The management CR is expected to report the status of **Warning** until the warm-standby management subsystem is deployed, and both management subsystems complete data replication. When you see the status message "Management is ready. HA Status Warning - see HAStatus in CR for details", you can move on to Installing API Connect on the warm-standby data center.

What to do next

Install API Connect in your warm-standby data center Installing API Connect on the warm-standby data center.

Deploy your Gateway and Analytics subsystems in your active data center. For shared namespace: Installing API Connect in a single namespace on OpenShift or multiple namespaces: Installing API Connect subsystems in different namespaces or environments on OpenShift.

Installing API Connect on the warm-standby data center

Add the **multiSiteHA** configuration to the API Connect Management and Portal YAML files before they are applied.

Before you begin

Verify that all the secrets, certificates, and issuers are ready:

```
oc get secrets -n <namespace>
ingress-ca                kubernetes.io/tls                     3      3d21h
mgmt-replication-client   kubernetes.io/tls                     3      3d21h
ptl-replication-client    kubernetes.io/tls                     3      3d21h

oc get certs -n <namespace>
mgmt-replication-client   True    mgmt-replication-client   3d21h   2024-08-17T13:04:27Z
ptl-replication-client    True    ptl-replication-client    3d21h   2024-08-17T13:04:26Z

oc get issuer -n <namespace>
ingress-issuer   True    3d21h
self-signed      True    3d21h
```

Where **<namespace>** are the namespaces you created for API Connect.
Note: If you have separate namespaces for Management and Portal subsystems, then run the check in both namespaces. You should not see the **mgmt-replication-client** in the portal namespace, nor the **ptl-replication-client** in the management namespace.

Procedure

1. Follow these steps for installing the Management subsystem on OpenShift Installing the Management subsystem in a shared namespace, but add the **encryptionSecret** and **multiSiteHA** section to the **mgmt_cr.yaml** file:

```
  siteName: <dc2 site name>
  encryptionSecret:
    secretName: mgmt-encryption-key
  multiSiteHA:
    mode: passive
    replicationEndpoint:
      annotations:
```

```
        cert-manager.io/issuer: ingress-issuer
      hosts:
      - name: mgmt-replication.<warm-standby data center ingress domain>
        secretName: mgmt-replication-server
    replicationPeerFQDN: mgmt-replication.<active data center ingress domain>
    tlsClient:
      secretName: mgmt-replication-client
```

where

- **<dc2 site name>** is the site name for this data center, as decided in Planning and initial preparation.
- **<active data center ingress domain>** and **<warm-standby data center ingress domain>** can be determined by running the command **oc get ingresses.config/cluster -o jsonpath={.spec.domain}** in each data center.

Note: If you are installing the Management subsystem in a separate namespace, these same steps apply.
Remember: Make sure that the management endpoints are set the same as for your active data center deployment.

2. Apply the update **mgmt_cr.yaml** file as directed in Installing the Management subsystem in a shared namespace. Monitor the deployment, but note that on the warm-standby there are fewer pods than on the active:

```
oc get ManagementCluster -n <namespace>

NAME          READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE      AGE
management    n/n     Running   10.0.7.0   10.0.7.0-1281        Management is ready. HA Status Warning - see HAStatus in CR
for details      62m
```

3. Follow these steps for installing the Portal subsystem on OpenShift Installing the Portal subsystem in a shared namespace, but add the **encryptionSecret** and **multiSiteHA** section to the **portal_cr.yaml** file.

```
  siteName: <dc2 site name>
  encryptionSecret:
    secretName: ptl-encryption-key
  multiSiteHA:
    mode: passive
    replicationEndpoint:
      annotations:
        cert-manager.io/issuer: ingress-issuer
      hosts:
      - name: ptl-replication.<warm-standby data center ingress domain>
        secretName: ptl-replication-server
    replicationPeerFQDN: ptl-replication.<warm-standby data center ingress domain>
    tlsClient:
      secretName: ptl-replication-client
```

where

- **<dc2 site name>** is the site name for this data center, as decided in Planning and initial preparation.
- **<active data center ingress domain>** and **<warm-standby data center ingress domain>** can be determined by running the command **oc get ingresses.config/cluster -o jsonpath={.spec.domain}** in each data center.

Remember: Make sure that the portal endpoints are set the same as for your active data center deployment.

4. Apply the updated **portal_cr.yaml** file as directed in Installing the Portal subsystem in a shared namespace.

```
oc get PortalCluster -n <namespace>
NAME      READY   STATUS    VERSION    RECONCILED VERSION   AGE
portal    6/6     Running   10.0.7.0   10.0.7.0-1281        47m
```

Results

While the management subsystems on the warm-standby and active data centers are synchronizing their databases, the management status reports **Warning**, and the **haStatus** reports **pending**:

```
oc get mgmt -n <namespace>

NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in CR
for details   8m59s

oc get mgmt -n <namespace> -o yaml

...
status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working, install or upgrade in progress.",
      "reason": "na",
      "status": "True",
      "type": "Pending"
    }
```

When the management database replication between sites is complete, the management status reports **Running**, and **status.haStatus** reports **Ready**:

```
oc get mgmt -n <namespace>

NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Running   10.0.7.0-0   10.0.7.0-0           Management is ready. HA status Ready - see HAStatus in CR for
details               8m59s

oc get mgmt -n <namespace> -o yaml
```

```
...
status:
  haStatus
  {
    "lastTransitionTime": "2023-03-31T19:47:08Z",
    "message": "Replication is working",
    "reason": "na",
    "status": "True",
    "type": "Ready"
  }
```

If the management database replication between sites fails for any reason other than because an install or upgrade is in progress, the `haStatus` output shows the reason, for example::

```
NAME          READY    STATUS     VERSION       RECONCILED VERSION    MESSAGE
AGE
management    n/n      Warning    10.0.7.0-0    10.0.7.0-0            Management is ready. HA Status Warning - see HAStatus in CR
for details    8m59s

status:
  haStatus
    {
      "lastTransitionTime": "2023-12-31T19:47:08Z",
      "message": "Replication not working",
      "reason": "na",
      "status": "True",
      "type": "PeerNotReachable"
   }
```

If the warning persists, see Troubleshooting a two data center deployment.
You can validate that your portal deployments are synchronizing by running **`oc get pods`** on both the active and warm-standby data centers. Confirm that the number and names of the pods all match (the UUIDs in the names might be different on each site), and that all are in the **`Ready`** state.

For additional replication verification checks, see Verifying replication between data centers. It is recommended to run a test failover, and confirm that all of the expected data is present and correct on the newly active site. See Failing over to the warm-standby.

What to do next

Deploy your Gateway and Analytics subsystems for shared namespace: Installing API Connect in a single namespace on OpenShift or multiple namespaces: Installing API Connect subsystems in different namespaces or environments on OpenShift

Configure your deployment: Cloud Manager configuration checklist, and complete a test failover to verify that your data replicated to the warm-standby. See Failing over to the warm-standby V10.0.6.0 for details.

Important: It is strongly recommended to take complete backups of both sites, for disaster recovery in case the API Connect installation in both data centers is lost, or if a replication failure occurs. Complete backups consist of your infrastructure configuration backup and your subsystem database backups. See Backup and restore requirements for a 2DCDR deployment.

Enable additional features post-install

After installing API Connect, you can enable additional features that are related to inter-subsystem communication.

- **Enable management CA verification on REST API calls**
  Enable the portal and gateway to validate the management subsystem's REST API server certificates.
- **Enable JWT security and disable mTLS between subsystems**
  Postinstallation steps to enable JWT security and disable mTLS between subsystems.

Enable management CA verification on REST API calls

Enable the portal and gateway to validate the management subsystem's REST API server certificates.

About this task

After you install API Connect, you can enable the portal and gateway to verify the REST API server certificate when they make calls to the REST API. The gateway and portal subsystems make calls to the management subsystem REST API during their normal operations. The gateway makes calls to the platform REST API, and the portal makes calls to the platform REST API and the consumer REST API. To verify the CA certificates of the REST API endpoints, enter the secret names of the CA certificates into the portal and gateway CRs.
Note:
If you are not sure of the secret names, follow these steps to determine their names:

1. Check the management subsystem CR with **`oc describe mgmt`**, and identify the issuer of the platform and consumer API endpoints:

   ```
   oc describe mgmt

   ...
   Platform API Endpoint:
       Annotations:
         cert-manager.io/issuer:               <instance name>-ingress-issuer
   ...
     Consumer API Endpoint:
   ```

```
    Annotations:
      cert-manager.io/issuer:               <instance name>-ingress-issuer
```

2. Describe the issuer with `oc describe issuer`, and identify the secret name:

```
oc describe issuer <instance name>-ingress-issuer
...
Spec:
  Ca:
    Secret Name:  <instance name>-ingress-ca
...
```

Note: If you are using `in-cluster` communication (see In-cluster service communication between subsystems), then the portal and gateway make REST calls on the service endpoints, instead of the `external` endpoints. In the steps documented in this topic, replace `mgmtPlatformEndpointCASecret` with `mgmtPlatformEndpointSvcCASecret`, and `mgmtConsumerEndpointCASecret` with `mgmtConsumerEndpointSvcCASecret`. For `secretName`, the default for the service endpoints is `mgmt-ca` or `<apic instance name>-mgmt-ca`.

Procedure

1. Update the portal CR.
   a. Edit the portal CR

      ```
      oc edit ptl
      ```

      Note: If you are using the OpenShift top-level CR deployment, edit your `APIConnectCluster` CR instead. Updates should be made in the `spec.portal` section.
   b. Set `spec.mgmtPlatformEndpointCASecret.secretName` and `spec.mgmtConsumerEndpointCASecret.secretName` to the name of the CA secret.

      ```
      spec:
      ...
        mgmtPlatformEndpointCASecret:
          secretName: <instance name>-ingress-ca
        mgmtConsumerEndpointCASecret:
          secretName: <instance name>-ingress-ca
      ```

2. Update the gateway CR.
   a. Edit the gateway CR

      ```
      oc edit gw
      ```

      Note: If you are using the OpenShift top-level CR deployment, edit your `APIConnectCluster` CR instead. Updates should be made in the `spec.portal` section.
   b. Set `spec.mgmtPlatformEndpointCASecret.secretName` to the name of the CA secret.

      ```
      spec:
      ...
        mgmtPlatformEndpointCASecret:
          secretName: <instance name>-ingress-ca
      ```

Enable JWT security and disable mTLS between subsystems

Postinstallation steps to enable JWT security and disable mTLS between subsystems.

About this task

JWT security is an alternative to using mTLS to secure inter-subsystem communication. For more information on JWT security and when to use it, see: Enable JWT security instead of mTLS.

If you disable mTLS then you must enable JWT.
Note: It is not possible to use JWT on the V5 compatible gateway to analytics message flow. This flow is secured by mTLS, which cannot be disabled.

Procedure

1. Describe the management CR to get the JWKS URL:

```
oc describe mgmt
```

```
Status:
  Endpoints:
    - name: jwksUrl
      secretName: api-endpoint
      type: API
      uri: https://api.apic.acme.com/api/cloud/oauth2/certs
```

2. For each portal, gateway, and analytics subsystem where you want to use JWT security, update the CR corresponding to that subsystem, and set `spec.mtlsValidateClient` to `false`, and `spec.jwksUrl` to the JWKS URL you identified in step 1.

```
spec:
...
  mtlsValidateClient: false
  jwksUrl: <JWKS URL>
```

Note: If you are using the OpenShift top-level CR deployment, edit your **APIConnectCluster** CR instead. Updates should be made in appropriate the **spec.<subsystem name>** section.

3. Enable JWT on the gateway to analytics communications flow. Enable the Use JWT switch for the registered gateway in the Topology page of the Cloud Manager UI.

Troubleshooting installation on OpenShift

Review the following known issues and troubleshooting tips if you encounter a problem while installing API Connect on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

V10.0.7.0: Failed integration-ibm-cloud-native-postgresql CatalogSource on ROKS 4.14 and OpenShift Container Platform 4.15

The API Connect operator creates the EDB catalog source in the same namespace as the API Connect operator.

The EDB catalog source reports a failed state and shows the following status message:

```
Status:
  Message:  couldn't ensure registry server - error ensuring pod: : error creating new pod: integration-ibm-cloud-native-
postgresql-: pods "integration-  ibm-cloud-native-postgresql-hnjbn" is forbidden: violates PodSecurity "restricted:v1.24":
allowPrivilegeEscalation != false (container "registry-server"   must set securityContext.allowPrivilegeEscalation=false),
unrestricted capabilities (container "registry-server" must set   securityContext.capabilities.drop=["ALL"]), runAsNonRoot !=
true (pod or container "registry-server" must set securityContext.runAsNonRoot=true),   seccompProfile (pod or container
"registry-server" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
  Reason:   RegistryServerError
```

The problem occurs when the namespace is set to enforce the restricted pod security admission policy with the **pod-security.kubernetes.io/enforce: restricted** label.

ROKS 4.14 and some OpenShift Container Platform versions such as 4.15 have **enforce** set to **restricted**.

To workaround this problem, update the **integration-ibm-cloud-native-postgresql CatalogSource** to use the restricted security context constraint:

```
oc patch CatalogSource integration-ibm-cloud-native-postgresql --type merge --patch '{"spec":{"grpcPodConfig":
{"securityContextConfig":"restricted"}}}'
```

One or more pods in CrashLoopBackoff or Error state, and report a certificate error in the logs

In rare cases, cert-manager might detect a certificate in a bad state right after it has been issued, and then re-issues the certificate. If a CA certificate has been issued twice, the certificate that was signed by the previously issued CA will be left stale and can't be validated by the newly issued CA. In this scenario, one of the following messages displays in the log:

- **javax.net.ssl.SSLHandshakeException: Received fatal alert: certificate_unknown**
- **Error: unable to verify the first certificate**
- **ERROR: openssl verify failed to verify the Portal CA tls.crt, ca.crt chain signed the Portal Server tls.crt cert**

Resolve the problem by completing the following steps:

1. Use **apicops** (v10 version 0.10.57+ required) to validate the certificates in the system:

   ```
   apicops upgrade:stale-certs -n <namespace>
   ```

2. If any certificate that is managed by cert-manager fails the validation, delete the stale certificate secret:

   ```
   oc delete secret <stale-secret> -n <namespace>
   ```

   Cert-manager automatically generates a new certificate to replace the one you deleted.

3. Use **apicops** to make sure all certificates can be verified successfully:

   ```
   apicops upgrade:stale-certs -n <namespace>
   ```

You see the denied: insufficient scope error during an air-gapped deployment

Problem: You encounter the **denied: insufficient scope** message while mirroring images during an air-gapped installation or upgrade.

Reason: This error occurs when a problem is encountered with the entitlement key that is used for obtaining images.

Solution: Obtain a new entitlement key by completing the following steps:

1. Log in to the IBM Container Library.
2. In the Container software library, select Get entitlement key.
3. After the Access your container software heading, click Copy key.
4. Copy the key to a safe location.

Apiconnect operator pod fails

Problem: During installation (or upgrade), the **apiconnect** operator fails with the following message:

```
panic: unable to build API support: unable to get Group and Resources: unable to retrieve the complete list of server APIs:
packages.operators.coreos.com/v1: the server is currently unable to handle the request

goroutine 1 [running]:
github.ibm.com/velox/apiconnect-operator/operator-utils/v2/apiversions.GetAPISupport(0x0)
        operator-utils/v2/apiversions/api-versions.go:89 +0x1e5
main.main()
        ibm-apiconnect/cmd/manager/main.go:188 +0x4ee
```

Additional symptoms:

- Apiconnect operator is in crash loopback status
- Kube **apiserver** pods log the following information:

  ```
  E1122 18:02:07.853093 18 available_controller.go:437] v1.packages.operators.coreos.com failed with:
   failing or missing response from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1:
   bad status from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1: 401
  ```

- The IP logged here belongs to the *package server* pod present in the **openshift-operator-lifecycle-manager** namespace
- Package server pods log the following error message: **/apis/packages.operators.coreos.com/v1 API call is being rejected with 401 issue**.

  ```
  E1122 18:10:25.614179 1 authentication.go:53] Unable to authenticate the request due to an error: x509:
  certificate signed by unknown authority I1122 18:10:25.614224 1 httplog.go:90]
  verb="GET" URI="/apis/packages.operators.coreos.com/v1" latency=161.243µs resp=401
  UserAgent="Go-http-client/2.0" srcIP="10.128.0.1:41370":
  ```

- The problem is intermittent

Solution:

- If you find the exact symptoms as described, the solution is to delete package server pods in the **openshift-operator-lifecycle-manager** namespace.
- New package server pods log the **200 Success** message for the same API call.

Disabling the Portal web endpoint check

When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the **portal-www** pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.
To disable the endpoint check, complete the following update:

On Kubernetes, OpenShift, and IBM® Cloud Pak for Integration

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

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

Upgrading API Connect

Upgrade API Connect on Kubernetes and OpenShift platforms to a newer version.

Upgrading your API Connect deployment consists of three phases:

1. Upgrade planning. Decide on your upgrade path based on your platform and current version. It might be necessary to upgrade your container environment before you can upgrade API Connect.

   Review any feature changes in the new API Connect version that might impact how you use API Connect and plan for such changes.

   Upgrade planning should be started as early as possible to identify any issues that might influence when you are able to upgrade, and any actions to take before upgrade.

2. Upgrade preparation. Run pre-upgrade checks on your current deployment, download the new version of API Connect from Fix Central, backup your current deployment. Upgrade your container platform if required.

   Upgrade preparation can be done at any time, and it is worth doing early to identify and fix any potential problems before your upgrade window. If you do complete the upgrade preparation early, repeat the pre-upgrade checks just before your upgrade window to check for any new problems.

3. Upgrade procedure. Updating your API Connect operator and operands to the new version. Upgrading cert-manager might also be necessary.

The upgrade process should be started only when all upgrade planning and preparation are completed. Interrupting and rolling back API Connect upgrades is not possible. Disaster recovery procedures must be followed to restore API Connect to the old version if problems are encountered during upgrade.

Important: API Connect v10.0.7.0 supports only backups to an S3 object-store for the management database. If your v10.0.6 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.
Upgrade to v10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store.

If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block. Support for SFTP and local backups will be provided in a future v10.0.x release.

The upgrade planning, preparation, and execution steps are different for each deployment type. Select from the following sections:

- **Upgrading on Kubernetes**
  Upgrade API Connect on Kubernetes distributions.
- **Upgrading on OpenShift and Cloud Pak for Integration**
  You can upgrade API Connect in an OpenShift environment and as part of IBM Cloud Pak for Integration.
- **Upgrading a two data center deployment on Kubernetes and OpenShift**
  How to upgrade a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift.
- **Upgrading a two data center deployment on Cloud Pak for Integration**
  How to upgrade a two data center disaster recovery (DR) deployment on Cloud Pak for Integration.

Upgrading on Kubernetes

Upgrade API Connect on Kubernetes distributions.

Note: For upgrading from v5, see Migrating a version 5 deployment to version 10.0.x.

- **Planning your API Connect upgrade on Kubernetes**
  Review the supported versions, requirements, and limitations for upgrading API Connect on Kubernetes.
- **Pre-upgrade preparation and checks on Kubernetes**
  Prepare your Kubernetes environment and API Connect deployment for upgrade.
- **Upgrading API Connect on Kubernetes**
  Upgrade your API Connect subsystems to a newer version on Kubernetes.
- **Troubleshooting upgrades on Kubernetes**
  Review the following troubleshooting guidance if you encounter a problem during an API Connect on Kubernetes.

Planning your API Connect upgrade on Kubernetes

Review the supported versions, requirements, and limitations for upgrading API Connect on Kubernetes.

Supported upgrade paths

Table 1. Supported upgrade paths on Kubernetes

Upgrade from:	How to upgrade to 10.0.7.0
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	Complete the procedure in Upgrading API Connect on Kubernetes.
10.0.5.2 or earlier	Complete two upgrades: 1. Upgrade to version 10.0.5.3 or later as explained in Upgrading on Kubernetes in the API Connect 10.0.5.x documentation. 2. Upgrade to the current release by completing the procedures in Upgrading API Connect on Kubernetes.

Attention: Different releases of API Connect support different versions of Kubernetes, so you might need to upgrade Kubernetes before you upgrade API Connect. For more information, see IBM API Connect Version 10 software product compatibility requirements.

Supported DataPower Gateway versions

You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

However, before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. When you find the information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

Supported versions of Kubernetes

Table 2 provides a quick reference of the supported Kubernetes versions for each API Connect release:

Table 2. API Connect and Kubernetes (K8S) compatibility matrix

API Connect version	K8S 1.24	K8S 1.25	K8S 1.26	K8S 1.27	K8S 1.28
10.0.5.3	✓	✓	✓		

API Connect version	K8S 1.24	K8S 1.25	K8S 1.26	K8S 1.27	K8S 1.28
10.0.5.4	✓	✓	✓	✓	
10.0.5.5			✓	✓	✓
10.0.6.0	✓	✓	✓	✓	
10.0.7.0			✓	✓	✓

When you upgrade API Connect, both your current deployment and your target release must support the same version of Kubernetes. After the API Connect upgrade, you can update Kubernetes to the highest version supported by the new release of API Connect.

It is possible that you must upgrade both Kubernetes and API Connect more than once to complete the process. For example, suppose that your current deployment is running API Connect 10.0.5.4 on Kubernetes v1.24 and you want to upgrade to API Connect 10.0.7.0. The highest version of Kubernetes that API Connect 10.0.5.4 supports is v1.26, and v1.26 is the lowest version of Kubernetes that API Connect 10.0.7.0 supports.

In this scenario, you must make multiple upgrades of both Kubernetes to reach the target versions; for example:

1. On your API Connect 10.0.5.4 deployment, upgrade Kubernetes from v1.24 to v1.25 and then to v1.26 (all interim upgrades are required).
2. Upgrade API Connect to 10.0.7.0 on Kubernetes v1.26.
3. Upgrade Kubernetes from v1.26 to v1.27 and then to v1.28 to reach the highest supported version of Kubernetes in API Connect 10.0.7.0.

Important upgrade considerations

Carefully review the following items, to plan and prepare for your upgrade.

API transaction failure during the upgrade of gateway
: The gateway subsystem remains available and continues processing API calls during the upgrade of the management, portal, and analytics subsystems. However, a few API transactions might fail during the upgrade of the gateway.

  When you upgrade a cluster of gateway pods, a few API transactions might fail. During the upgrade, Kubernetes removes the pod from the load balancer configuration, deletes the pod, and then starts a new pod. The steps are repeated for each pod. Socket hang-ups occur on transactions that are in process at the time the pod is terminated.

  The number of transactions that fail depends on the rate of incoming transactions, and the length of time that is needed to complete each transaction. Typically the number of failures is a small percentage. This behavior is expected during an upgrade. If the failure level is not acceptable, schedule the upgrade during an off-hours maintenance window.

  DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when you upgrade. Workloads with long-lived connections are more vulnerable to failed API transactions during upgrading.

  You can limit the number of failed API transactions during the upgrade by using the DataPower Operator's **`lifecycle`** property to configure the **`preStop`** container lifecycle hook on the gateway pods. This approach mitigates the risk of API failures during the rolling update of the gateway StatefulSet by sleeping the pod for a span of time, allowing in-flight transactions to complete before the SIGTERM is delivered to the container. While this feature does not guarantee no in-flight API call failures, it does provide some mitigation for in-flight transactions that can complete successfully within the configured time window. For more information, see Delaying SIGTERM with preStop in the DataPower documentation.

FIPS support
: API Connect is not supported in a FIPS-enabled environment on Kubernetes.

OAuth provider resources with **`api.securityDefintions`**
: If you have native OAuth providers that are configured with the **`api.securityDefintions`** field assigned, then upgrade fails. Before upgrade, remove all **`api.securityDefintions`** from all native OAuth providers that you configured in the Cloud Manager and API Manager UIs:

  - Configuring a native OAuth provider in the Cloud Manager UI
  - Configuring a native OAuth provider in the API Manager UI

  Switch to the source view and delete any **`securityDefintions`** sections that are present.

SFTP management backups (V10.0.6.0 to V10.0.7.0 upgrade)
: API Connect V10.0.7.0 supports only backups to an S3 object-store for the management database. If your V10.0.6.0 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.

  Upgrade to V10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store.

  If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block. Support for SFTP and local backups will be provided in a future V10.0.x release.

Extra backup folder appended to management database backup path
: When you upgrade to V10.0.7.0 or later with S3 backups configured, an extra backup path folder /edb is appended to your backup path. The /edb path is to distinguish new EDB backups from your previous management backups.

Extra persistent storage requirement for management subsystem
: Upgrade to V10.0.7.0 or later requires extra persistent storage space for the management subsystem. Check that your environment has at least as much extra capacity as the capacity assigned to your existing Postgres PVC:

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

  ```
  NAME                                     STATUS   VOLUME               CAPACITY   ACCESS MODES   STORAGECLASS    AGE
  def-management-site1-postgres            Bound    local-pv-7b528204    244Gi      RWO            local-storage   125d
  def-management-site1-postgres-pgbr-repo  Bound    local-pv-8d425510    244Gi      RWO            local-storage   125d
  def-management-site1-postgres-wal        Bound    local-pv-ee8feb38    244Gi      RWO            local-storage   125d
  ```

Analytics persistent queue enablement requires extra PV. Upgrades from V10.0.6 and V10.0.5
: The analytics persistent queue feature is enabled during the upgrade (if it was not already enabled). This feature requires an extra Physical Volume (PV). Before you upgrade from V10.0.6 or V10.0.5.x, ensure that your environment has an available PV.

Pre-upgrade preparation and checks on Kubernetes

Prepare your Kubernetes environment and API Connect deployment for upgrade.

This topic covers all the preparation steps you must complete before you can start the upgrade process.

Before following the steps in this topic, review Planning your API Connect upgrade on Kubernetes to ensure that you are following a supported upgrade path and that you are aware of any API Connect feature changes that might affect your deployment.

The pre-upgrade health checks that are documented in this topic should always be run just before you start the upgrade process, but you can also run them at any time.
Tip: Run the pre-upgrade health checks during your upgrade planning stage to identify any problems as early as possible. Run the checks again just before upgrade to confirm API Connect is still ready for upgrade.

Gathering and preparing upgrade files

1. Obtain the API Connect files from IBM Fix Central.
   From the IBM Fix Central site, download the following files:

   IBM® API Connect *<version>* for Containers
   Docker images for all API Connect subsystems. Filename is apiconnect-image-tool_*<version>*.
   IBM® API Connect *<version>* Operator Release Files for Containers
   Kubernetes operators and API Connect Custom Resource (CR) templates. Filename is apiconnect-operator-release-files_*<version>*.
   IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
   Toolkit command-line utility. Packaged as a stand-alone file, or with API Designer or Loopback:
   - IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
   - IBM® API Connect *<version>* Toolkit with Loopback for *<operating_system_type>*
   - IBM® API Connect *<version>* Toolkit Designer with Loopback for *<operating_system_type>*

   Filename is toolkit-...*<operating system>*_*<version>*.
   Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See Installing the toolkit.

   IBM® API Connect *<version>* Local Test Environment
   Optional test environment. See Testing an API with the Local Test Environment
   Filename is apic-lte-*<operating system>*_*<version>*.

   IBM® API Connect *<version>* Security Signature Bundle File
   Checksum files that you can use to verify the integrity of your downloads.
   Filename is signatures_*<version>*.

   To verify the files against the signatures, see Verifying the integrity of IBM product files.

2. Upload the image files that you obtained in step 1 to your docker registry.
   a. Load the image-tool image for the new API Connect version into your local Docker registry:

      ```
      docker load < apiconnect-image-tool-<version>.tar.gz
      ```

      Ensure that the registry has sufficient disk space for the files.

   b. If your Docker registry requires repositories to be created before images can be pushed, create the repositories for each of the images that the image tool lists. If your Docker registry does not require creation of repositories, skip to step 2.c.
      i. Run the following command to get a list of the images from image-tool:

         ```
         docker run --rm apiconnect-image-tool-<version> version --images
         ```

      ii. From the output of each entry of the form **<image-name>:<image-tag>**, use your Docker registry repository create command to create a repository for **<image-name>**.
      For example, in AWS ECR the command for each **<image-name>** is:

         ```
         aws ecr create-repository --repository-name <image-name>
         ```

   c. Upload the image:
      - If you do not need to authenticate with the docker registry, use:

        ```
        docker run --rm apiconnect-image-tool-<version> upload <registry-url>
        ```

      - If you need to authenticate and your docker registry accepts the username and password arguments, use:

        ```
        docker run --rm apiconnect-image-tool-<version> upload <registry-url> --username <username> --password <password>
        ```

      - Otherwise, if you need the image-tool to use your local Docker credentials (for example, with IBM Container Registry), first authenticate with your Docker registry, and then upload images with the command:

        ```
        docker run --rm -v ~/.docker:/root/.docker --user 0 apiconnect-image-tool-<version> upload <registry-url>
        ```

        Review the following installation notes for your environment:

        - Alternative Docker command for macOS if Docker is configured to use osxkeychain
        - Docker authentication notes
        - Providing a certificate for verification, or disabling TLS verification

      If necessary, you can populate a remote container registry with repositories. Then you can push the images from the local registry to the remote registry.

3. Extract the **IBM API Connect *<version>* Operator Release Files for Containers** that you downloaded in step 1.
   Create a directory called **helper_files** and extract the contents of **helper_files.zip** from the release_files.zip into the **helper_files** directory.

Pre-upgrade health-checks

1. Check that the API Connect operator and subsystems are all running.
   To verify the health of each subsystem, run the following commands:

   ```
   kubectl get ManagementCluster -n <management namespace>
   kubectl get GatewayCluster -n <gateway namespace>
   kubectl get PortalCluster -n <portal namespace>
   kubectl get AnalyticsCluster -n <analyticsnamespace>
   ```

   Confirm that all pods are **READY**, for example:

   ```
   kubectl get PortalCluster -n apic
   NAME      READY   STATUS    VERSION    RECONCILED VERSION   AGE
   portal    3/3     Running   10.0.7.0   10.0.7.0-95    57m
   ```

2. Verify that the management subsystem's **pgcluster** is working correctly:
   a. Get the name of the **pgcluster**:

      ```
      kubectl get pgcluster -n <management namespace>
      ```

      The response displays the name of the postgres cluster that is running in the specified namespace.

   b. Check the status of the **pgcluster**:

      ```
      kubectl get pgcluster <pgcluster_name> -n <APIC_namespace> -o yaml | grep status -A 2 | tail -n3
      ```

      A successful response looks like the following example, where the state is **Initialized**:

      ```
      status:
          message: Cluster has been initialized
          state: pgcluster Initialized
      ```

      Important: If the **pgcluster** returns any other state, then do not proceed with the upgrade.
      - If backup or restore jobs are running, wait until they complete and then check the status again. Do not proceed with the upgrade until the status is **Initialized**.
      - If all background jobs are complete but the **pgcluster** does not return **state: pgcluster Initialized**, then contact IBM Support for assistance.
3. If you used any microservice image overrides in any of your API Connect subsystem CRs, remove the image overrides before upgrade.
4. Run the pre-upgrade health check:
   Attention: The pre-upgrade health check is a required step. Failure to run this check might result in upgrade failure that requires disaster recovery to restore your API Connect deployment.
   Note: If you have a 2DCDR deployment, run this check only on the active management subsystem.
   a. Verify that the **apicops** utility is installed and that you have the latest version by running the following command:

      ```
      apicops --version
      ```

      For more information about the **apicops** utility, see The API Connect operations tool: apicops.

   b. Run the following command to set the KUBECONFIG environment variable.

      ```
      export KUBECONFIG=</path/to/kubeconfig>
      ```

   c. Run the **apicops pre-upgrade** check:

      ```
      apicops version:pre-upgrade -n <namespace>
      ```

      If your API Connect deployment is working correctly, the output shows no error messages.

      Note: The **apicops pre-upgrade** command asks for the Cloud Manager admin password so that it can use the platform API.
      If you do not have this password, or the platform API URL is not accessible from your **apicops** location, then this part of the pre-upgrade check can be skipped by adding the argument **--no-topology**.

      The topology check output that the API call produces is typically only required for debugging purposes, when the **apicops version:pre-upgrade** script identifies problems with your installation.

Additional pre-upgrade checks and operations

Extra checks and operations to review and complete before you start the upgrade.

1. If you are upgrading to V10.0.7.0, and you do not have management database backups configured, or you have them configured for SFTP or local, then change your management database backup settings to S3. See Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0.

   V10.0.7.0 supports only backups to an S3 object-store. If you are not able to configure an S3 object-store, then to upgrade to v10.0.7.0 you must delete the **databaseBackup** section from your management CR, and add the **apiconnect-operator/backups-not-configured: "true"** annotation before you upgrade.

2. If you are upgrading to V10.0.7.0, and the analytics persistent queue is not enabled, check that you have an extra Physical Volume available.
   For more information on this requirement, see Analytics persistent queue enablement requires extra PV.

3. If you used any microservice image overrides in any of your API Connect subsystem CRs, remove the image overrides before upgrade.

Backup your API Connect deployment

If upgrade fails, rollback is not possible. Before upgrade complete the disaster recovery preparation for all your subsystems.

For instructions on backing up and disaster recovery preparation, see the following topics:

- v10.0.7.0 and later: Backing up, restoring, and disaster recovery.
- v10.0.6.0: Backups on Kubernetes V10.0.6.0, Disaster recovery on Kubernetes v10.0.6.0.
- v10.0.5.x: Backups on Kubernetes, Disaster recovery.

Upgrading API Connect on Kubernetes

Upgrade your API Connect subsystems to a newer version on Kubernetes.

Before you begin

Review Planning your API Connect upgrade on Kubernetes and complete the Pre-upgrade preparation and checks on Kubernetes.

About this task

- To ensure operator compatibility, upgrade the API Connect management subsystem first and the gateway subsystem second. This requirement applies to all upgrade scenarios.
- The gateway subsystem remains available during the upgrade of the management, portal, and analytics subsystems.

Notes:

- If a backup is running or is scheduled to run within a few hours, do not start the upgrade process.
- Do not do maintenance tasks such as rotating key-certificates, restoring from a backup, or starting a new backup, at any time while the upgrade process is running.

Procedure

1. If you have not run the pre-upgrade health checks recently, repeat these checks.
2. Apply the API Connect CRDs that you downloaded and extracted in the pre-upgrade steps:

   ```
   kubectl apply -f ibm-apiconnect-crds.yaml
   ```

3. Apply the new DataPower Operator YAML into the namespace where the DataPower operator is running.
   a. Edit the `ibm-datapower.yaml` file and set the following properties:the `namespace` property to the namespace where you deployed the operator.
      - Set all instances of the `namespace` property to the namespace where you deployed the operator.
      - Set the `containers.image` property to the location of the `datapower-operator` image, either uploaded to your own registry or pulled from a public registry.

   b. Run the following command:

      ```
      kubectl -n <namespace> apply -f ibm-datapower.yaml
      ```

      The gateway CR goes to `Pending` state when the operator is updated, and then changes to `Running` after installation of the API Connect operator in the next step.

      Troubleshooting: If your DataPower operator pod fails to start, see Troubleshooting upgrade.

4. Upgrade cert-manager to version 1.11.5.
   Note: If you are upgrading from 10.0.5.5, then cert-manager 1.11.5 is already installed and you can skip this step.
   a. Run the following command to back up the existing certificates and issuers to a file called backup.yaml:

      ```
      kubectl get --all-namespaces -oyaml issuer,clusterissuer,cert,secret > backup.yaml
      ```

   b. Run the following command to verify the version of cert-manager that you are upgrading:
      - API Connect 10.0.5.3: verify that you are upgrading cert-manager 1.7.1:

        ```
        kubectl get crds certificates.cert-manager.io -o jsonpath='{.metadata.labels.app\.kubernetes\.io\/version}'
        v1.7.1
        ```

      - API Connect 10.0.5.4 or 10.0.6.0: verify that you are upgrading cert-manager 1.9.1:

        ```
        kubectl get crds certificates.cert-manager.io -o jsonpath='{.metadata.labels.app\.kubernetes\.io\/version}'
        v1.9.1
        ```

   c. Run the following command to upgrade cert-manager to version 1.11.5:

      ```
      kubectl apply -f helper_files/cert-manager-1.11.5.yaml
      ```

5. Clean up the webhook configuration before you deploy the newer API Connect operator:
   a. Run the following command to get the webhook configuration:

      ```
      kubectl get mutatingwebhookconfiguration,validatingwebhookconfiguration | grep ibm-apiconnect
      ```

   b. Use the `kubectl delete` command to delete the webhook configuration; for example:

      ```
      kubectl delete mutatingwebhookconfiguration ibm-apiconnect-mutating-webhook-configuration
      kubectl delete validatingwebhookconfiguration ibm-apiconnect-validating-webhook-configuration
      ```

6. If you are upgrading to v10.0.7.0 and you have management database backups set to SFTP or local, then change your management database backup settings to S3.
   See Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0.
   V10.0.7.0 supports only backups to an S3 object-store. If you are not able to configure an S3 object-store, then to upgrade to v10.0.7.0 you must delete the `databaseBackup` section from your management CR, and add the `apiconnect-operator/backups-not-configured: "true"` annotation before you upgrade, see step 1.d on Upgrading the management subsystem on Kubernetes.

7. Apply the new API Connect operator YAML into the namespace where the API Connect operator is running.

- For a single-namespace deployment:
  - If the operator is not running in the `default` namespace, edit the `ibm-apiconnect.yaml` and replace all references to `default` with the name of the namespace where you deployed API Connect.
    Note: If you are using Operator Lifecycle Manager (OLM), then skip this step.
  - Edit `ibm-apiconnect.yaml` file and replace the value of each `image` property with the location of the `apiconnect` operator images (from the `ibm-apiconnect` container and the `ibm-apiconnect-init` container), either uploaded to your own registry or pulled from a public registry.
  - Run the following command:

    ```
    kubectl -n <namespace> apply -f ibm-apiconnect.yaml
    ```

- For a multi-namespace deployment:
  - Edit the `ibm-apiconnect-distributed.yaml` file and replace each occurrence of `$OPERATOR_NAMESPACE` with the namespace for the deployment.
  - Also in `ibm-apiconnect-distributed.yaml`, set the values of the `containers.image` properties. Replace the placeholder values `REPLACE-DOCKER-REGISTRY` with the docker registry host location of the API Connect operator image (either uploaded to own registry or pulled from public registry).
  - Install `ibm-apiconnect-distributed.yaml` with the following command:

    ```
    kubectl apply -f ibm-apiconnect-distributed.yaml
    ```

8. Verify that the `ibm-datapower-operator` and the `ibm-apiconnect` operators are restarted.
9. Ensure that the `apiconnect` operator re-created the necessary microservices:

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

   Do not proceed to upgrade the API Connect subsytems until the operator upgrade is complete.

10. Upgrade the API Connect subsystems:
    a. Upgrading the management subsystem on Kubernetes
    b. Upgrading the portal subsystem on Kubernetes
    c. Upgrading the analytics subsystem on Kubernetes
    d. Upgrading the gateway subsystem on Kubernetes
11. Verify that the upgraded subsystems report as `Running` and the `RECONCILED VERSION` displays the new version of API Connect.
    Run the following command:

    ```
    kubectl get apic --all-namespaces
    ```

    Example response:

    ```
    NAME                                                READY   STATUS    VERSION              RECONCILED VERSION      AGE
    analyticscluster.analytics.apiconnect.ibm.com/analytics      8/8     Running   10.0.7.0   10.0.7.0-1074   121m

    NAME                                       PHASE     READY   SUMMARY                            VERSION    AGE
    datapowerservice.datapower.ibm.com/gw1   Running   True    StatefulSet replicas ready: 1/1   10.0.7.0   100m

    NAME                                       PHASE     LAST EVENT   WORK PENDING   WORK IN-PROGRESS   AGE
    datapowermonitor.datapower.ibm.com/gw1   Running                  false          false              100m

    NAME                                          READY   STATUS    VERSION              RECONCILED VERSION      AGE
    gatewaycluster.gateway.apiconnect.ibm.com/gw1   2/2     Running   10.0.7.0   10.0.7.0-1074  100m

    NAME                                             READY   STATUS    VERSION              RECONCILED VERSION      AGE
    managementcluster.management.apiconnect.ibm.com/m1   16/16   Running   10.0.7.0   110.0.7.0-1074   162m


    NAME                                           READY   STATUS    VERSION              RECONCILED VERSION      AGE
    portalcluster.portal.apiconnect.ibm.com/portal   3/3     Running   10.0.7.0   10.0.7.0-1074   139m
    ```

    Important: If you need to restart the deployment, wait until all portal sites complete the upgrade.
    After the portal subsystem upgrade is complete, each portal site is upgraded. You can monitor the site upgrade progress from the `MESSAGE` column in the `oc get ptl` output. You can still use the portal while sites are upgrading, although a maintenance page is shown for any sites that are being upgraded. When the site upgrades are complete, the `oc get ptl` output shows how many sites the portal is serving:

    ```
    NAME      READY   STATUS    VERSION          RECONCILED VERSION   MESSAGE              AGE
    portal    3/3     Running   <version>        <version>            Serving 2 sites      22h
    ```

    On two data center disaster recovery deployments, the sites are not upgraded until both data centers are upgraded.

12. If you are upgrading from v10.0.5.3, update the certificates with the ingress-issuer-v1.yaml from your target release.

    ```
    kubectl apply -f helper_files/ingress-issuer-v1.yaml -n <namespace>
    ```

    For a multi-namespace deployment, repeat the command for each namespace.

13. Optional: For the optional components API Connect Toolkit and API Connect Local Test Environment, install the latest version of each after you complete the upgrade of the subsystems.

- **Upgrading the management subsystem on Kubernetes**
  Upgrade the management subsystem to a newer version of API Connect on Kubernetes.
- **Upgrading the portal subsystem on Kubernetes**
  Upgrade the portal subsystem to a newer version of API Connect on Kubernetes.
- **Upgrading the analytics subsystem on Kubernetes**
  Upgrade the analytics subsystem to a newer version of API Connect on Kubernetes.
- **Upgrading the gateway subsystem on Kubernetes**
  Upgrade the gateway subsystem to a newer version of API Connect on Kubernetes.

Upgrading the management subsystem on Kubernetes

Upgrade the management subsystem to a newer version of API Connect on Kubernetes.

Before you begin

Review and complete all steps in:

- Planning your API Connect upgrade on Kubernetes
- Pre-upgrade preparation and checks on Kubernetes
- Upgrading API Connect on Kubernetes

About this task

Upgrade the management subsystem by updating the version property in your management CR. The API Connect operator detects this change and starts the upgrade.

Procedure

1. Update the management CR for the new version of API Connect.
   a. Run the following command to edit the management CR:

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

   b. Update `spec.version` in the management CR to your target version:
      For example:

      ```
      version: 10.0.7.0
      ```

   c. If you are upgrading to a version of API Connect that requires a new license, update `spec.license`.
      For example:

      ```
      license: L-MMBZ-295QZQ
      ```

      For the list of licenses, see API Connect licenses.

   d. If you are upgrading from v10.0.6.0 to v10.0.7.0 and were not able to configure an S3 object-store for your management database backups, then remove the `databaseBackup` section, and add the following annotation:

      ```
      metadata:
        annotations:
          apiconnect-operator/backups-not-configured: "true"
      ```

      If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
   e. If you have a two data center disaster recovery installation, and are upgrading the warm-standby to V10.0.7.0 or later, then add the annotation:

      ```
      apiconnect-operator/dr-warm-standby-upgrade-data-deletion-confirmation: "true"
      ```

      For more information about two data center disaster recovery upgrade, see Upgrading a two data center deployment on Kubernetes and OpenShift.
   f. Run the following command to save and apply the CR: `wq`
      When you save the updated CR, the upgrade starts automatically.

      Troubleshooting: If you see an error message when you attempt to save the CR, see the known upgrade issues.

2. Run the following command to verify that the upgrade was successful:

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

   Example output after management subsystem upgrade:

   ```
   NAME                                                READY   STATUS    VERSION              RECONCILED VERSION      AGE
   managementcluster.management.apiconnect.ibm.com/m1   16/16   Running   <version>   <version-build>   162m

   NAME                                                                  STATUS    MESSAGE
   AGE
   managementdbupgrade.management.apiconnect.ibm.com/management-up-8c28g   Running   Upgrade is running (DB Schema/data are
   up-to-date)   13m
   ```

3. Restart all nats-server pods by running the following command:

   ```
   kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
   ```

What to do next

Upgrade your portal subsystem: Upgrading the portal subsystem on Kubernetes.

Upgrading the portal subsystem on Kubernetes

Upgrade the portal subsystem to a newer version of API Connect on Kubernetes.

Before you begin

Review and complete all steps in:

- Planning your API Connect upgrade on Kubernetes
- Pre-upgrade preparation and checks on Kubernetes
- Upgrading API Connect on Kubernetes
- Upgrading the management subsystem on Kubernetes

About this task

Upgrade the portal subsystem by updating the version property in your portal CR. The API Connect operator detects this change and starts the upgrade.

Procedure

1. Update the portal CR for the new version of API Connect.
   a. Run the following command to edit your portal CR:

      ```
      kubectl -n <namespace> edit ptl
      ```

   b. Set the `spec.version` property in the portal CR to your target version:
      For example:

      ```
      version: 10.0.7.0
      ```

   c. If you are upgrading to a version of API Connect that requires a new license, update `spec.license`.
      For example:

      ```
      license: L-MMBZ-295QZQ
      ```

      For the list of licenses, see API Connect licenses.

   d. Run the following command to save and apply the CR: `wq`
      When you save the updated CR, the upgrade starts automatically.

      Troubleshooting: If you see an error message when you attempt to save the CR, see the known upgrade issues.

2. Run the following command to verify that the upgrade was successful:

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

   Example output after portal subsystem upgrade:

   ```
   NAME                                                READY   STATUS    VERSION     RECONCILED VERSION   MESSAGE
   AGE
   portalcluster.portal.apiconnect.ibm.com/portal      3/3     Running   <version>   <version>            Site Upgrades
   Executing:(1) Queued(5) Failed(0) Success(0)   22h
   ```

   Note: After the portal subsystem upgrade is complete, each portal site is upgraded. You can monitor the site upgrade progress from the `MESSAGE` column in the `kubectl get apic` output. You can still use the portal while sites are upgrading, although a maintenance page is shown for any sites that are being upgraded. When the site upgrades are complete, the `kubectl get apic` output shows how many sites the portal is serving:

   ```
   NAME      READY   STATUS    VERSION          RECONCILED VERSION   MESSAGE              AGE
   portal    3/3     Running   <version>        <version>            Serving 2 sites      22h
   ```

   On two data center disaster recovery deployments, the sites are not upgraded until both data centers are upgraded.

Upgrading the analytics subsystem on Kubernetes

Upgrade the analytics subsystem to a newer version of API Connect on Kubernetes.

Before you begin

Review and complete all steps in:

- Planning your API Connect upgrade on Kubernetes
- Pre-upgrade preparation and checks on Kubernetes
- Upgrading API Connect on Kubernetes
- Upgrading the management subsystem on Kubernetes

About this task

Upgrade the analytics subsystem by updating the version property in your analytics CR. The API Connect operator detects this change and starts the upgrade.

Procedure

1. Update the analytics CR for the new version of API Connect.
   a. Run the following command to edit the CR:

```
kubectl -n <namespace> edit a7s
```

b. Set the `spec.version` property in the analytics CR to your target version:
For example:

```
version: 10.0.7.0
```

c. If you are upgrading to a version of API Connect that requires a new license, update `spec.license`.
For example:

```
license: L-MMBZ-295QZQ
```

For the list of licenses, see API Connect licenses.

d. Save your changes to the CR.
When you save the updated CR, the upgrade starts automatically.

Troubleshooting: If you see an error message when you attempt to save the CR, see the known upgrade issues.

2. Run the following command to verify that the upgrade was successful:

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

Example output after an analytics subsystem upgrade:

```
NAME                                                  READY   STATUS    VERSION          RECONCILED VERSION   AGE
analyticscluster.analytics.apiconnect.ibm.com/analytics   5/5     Running   <version>   <version-build>   121m
```

Upgrading the gateway subsystem on Kubernetes

Upgrade the gateway subsystem to a newer version of API Connect on Kubernetes.

Before you begin

Review and complete all steps in:

- Planning your API Connect upgrade on Kubernetes
- Pre-upgrade preparation and checks on Kubernetes
- Upgrading API Connect on Kubernetes
- Upgrading the management subsystem on Kubernetes

Note: To upgrade a high-availability cluster, ensure that you meet the following requirements:

- Gateways must be updated one at a time.
- Before you start the upgrade, a single gateway must be running as primary for all gateway-peering definitions.
- To upgrade multiple gateways, upgrade the primary gateway last.
Ensure that the pod with a name like `gwv6-0` or `gwv5-0` is the primary because it is the last node to be upgraded.

  To determine which gateway is running as primary, use either
  - The **show gateway-peering-status** command in the DataPower CLI. Access the CLI with:

    ```
    kubectl attach -it <gateway pod name>
    ```

  - The Gateway Peering Status display in the WebGUI (in the API Connect application domain).

  To move the primary to the DataPower on which you are working, you can issue the following command:

```
gateway-peering-switch-primary <peering-object-name>
```

About this task

Upgrade the analytics subsystem by updating the version property in your analytics CR. The API Connect operator detects this change and starts the upgrade.

Gateway upgrade can result in failure of some in-flight transactions, for more information see: API transactions might fail during the upgrade of gateway pods.

Procedure

1. Update the gateway CR for the new version of API Connect.
   a. Run the following command to edit the CR:

   ```
   kubectl -n <namespace> edit gw <gateway CR name>
   ```

   where *<gateway CR name>* is the name of your gateway CR. The names of your gateway CRs are shown in the output of: `kubectl get gw`.

   b. Set the `spec.version` property in the gateway CR to your target version:
   For example:

   ```
   version: 10.0.7.0
   ```

   c. If you are upgrading to a version of API Connect that requires a new license, update `spec.license`.
   For example:

   ```
   license: L-MMBZ-295QZQ
   ```

For the list of licenses, see API Connect licenses.

d. Delete any `template` or `dataPowerOverride` override sections.
If the CR contains an override, an upgrade is not possible.

e. Run the following command to save and apply the CR: `wq`
When you save the updated CR, the upgrade starts automatically.

Troubleshooting: If you see an error message when you attempt to save the CR, see the known upgrade issues.

2. Run the following command to verify that the upgrade was successful:

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

Example output after a gateway subsystem upgrade:

```
NAME                                          READY   STATUS    VERSION              RECONCILED VERSION      AGE
gatewaycluster.gateway.apiconnect.ibm.com/gw1   2/2     Running   <version>  <version-build>  100m
```

Troubleshooting upgrades on Kubernetes

Review the following troubleshooting guidance if you encounter a problem during an API Connect on Kubernetes.

Subsystem is stuck in `Pending` state with a reason of `PreUpgradeCheckInProgress`

Before the subsystem microservices are upgraded, the operator triggers a set of pre-upgrade checks that must pass for the upgrade to proceed. If one or more of the checks fail, the subsystem status remains in `Pending` state with a reason of `PreUpgradeCheckInProgress`. Check the status condition of the subsystem CR to confirm the pre-upgrade check failed. The `status.PreUpgradeCheck` property contains a summary of the failed checks. Full logs for the checks that are carried out can be viewed in the `ConfigMap` referenced in the `status.PreUpgradeCheck` property. The pre-upgrade checks automatically retry until they successfully pass. If you are unable to rectify the problem that causes a check to fail, then open an IBM support case.

License webhook error

If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

```
admission webhook "vmanagementcluster.kb.io" denied the request:
ManagementCluster.management.apiconnect.ibm.com "management" is invalid:
spec.license.license: Invalid value: "L-RJON-BYGHM4":
License L-RJON-BYGHM4 is invalid for the chosen version 10.0.7.0.
Please refer license document https://ibm.biz/apiclicenses
```

To resolve the error, see API Connect licenses for the list of the available license IDs and select the appropriate license IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.

Taskmanager error syncing management and gateway

If the gateways are not in sync with management after upgrade, check if the management subsystem `taskmanager` pods log the following error message. It starts 15 minutes after upgrade and repeats every 15 minutes for any stuck task.

```
TASK: Stale claimed task set to errored state:
```

If these errors are reported, restart all the `management-natscluster` pods, for example: `management-natscluster-1`.

```
kubectl -n <namespace> delete pod management-natscluster-1 management-natscluster-2 management-natscluster-3
```

DataPower operator fails to start

There is a known issue on Kubernetes that can cause the DataPower operator to fail to start. In this case, the DataPower operator pods can fail to schedule, and display the status message: `no nodes match pod topology spread constraints (missing required label)`. For example:

```
0/15 nodes are available: 12 node(s) didn't match pod topology spread constraints (missing required label),
3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate.
```

You can workaround the issue by editing the DataPower operator deployment and reapplying it, as follows:

1. Delete the DataPower operator deployment, if deployed already:

```
kubectl delete -f ibm-datapower.yaml -n <namespace>
```

2. Open `ibm-datapower.yaml`, and locate the `topologySpreadConstraints:` section. For example:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: zone
  whenUnsatisfiable: DoNotSchedule
```

3. Replace the values for `topologyKey:` and `whenUnsatisfiable:` with the corrected values that are shown in the following example:

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: ScheduleAnyway
```

4. Save `ibm-datapower.yaml` and deploy the file to the cluster:

   ```
   kubectl apply -f ibm-datapower.yaml -n <namespace>
   ```

V10.0.6.0: Postgres error message when you save the updated management CR

For upgrades to v10.0.6.0 only.

If you see an error message when you attempt to save the CR, check if it is one of the following known issues:

- Webhook error: `Original postgres primary is running as replica`
  If you see the error message:

  ```
  Original PostgreSQL primary is running as replica,
  please perform failover to original primary before attempting upgrade. Refer - https://ibm.biz/BdPLWU"
  ```

  you must complete a Postgres failover before the upgrade can proceed, see Postgres failover steps. After you apply the Postgres failover steps, the upgrade resumes automatically.
- Webhook error: `Original PostgreSQL primary not found`
  If you see the error message:

  ```
  Original PostgreSQL primary not found. Upgrade is blocked.
  Set apiconnect-operator/db-primary-not-found-allow-upgrade=true annotation to unblock upgrade.
  Perform db backup and restore after upgrade to restore original primary.
  ```

  take the following actions to complete the upgrade and fix the cause of the error message:
  1. Edit the management CR:

     ```
     kubectl -n $NAMESPACE edit mgmt <CLUSTERNAME>
     ```

  2. Add the following annotation:

     ```
     apiVersion: management.apiconnect.ibm.com/v1beta1
     kind: ManagementCluster
     metadata:
       annotations:
         apiconnect-operator/db-primary-not-found-allow-upgrade=true
         ...
     ```

  3. Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

     ```
     Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
     ```

  4. Take a new management database backup.
  5. Immediately restore from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.

Unexpected behavior in Cloud Manager and API Manager UIs after upgrade

Stale browser cache issues can cause problems after an upgrade. To remedy this problem, clear your browser cache, and open a new browser window.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

Portal sites failed to be upgraded successfully

If one or more of the Portal sites failed to be upgraded successfully, check the portal-www admin container logs to see what prevented the site upgrade from completing successfully.

To trigger the site upgrade again, exec into the portal-www pod admin container and run the following command:

```
upgrade_devportal -s <site_uuid> -p <platform>
```

where:

- *<site_uuid>* can be obtained by running the command: `list_sites`
- *<platform>* can be obtained by running the command: `list_platforms`

- **Enabling gateway peering and verifying cluster status**
  Scale the replicas to 0 and back up to reset gateway-peering.

Enabling gateway peering and verifying cluster status

Scale the replicas to 0 and back up to reset gateway-peering.

Before you begin

This task is only needed under certain scenarios, such as when you rotate the root CA, and is not automatically required for an upgrade.

About this task

Follow these steps to fix gateway peering. When the ingress issuer changes, the gateway pods must be scaled down and back up. This process causes 5 to 10 minutes of API downtime.

Procedure

1. After operand upgrade completes, wait for management, portal, and analytics subsystems to report `Running` status.
2. Scale down the gateway firmware containers by editing the gateway CR and setting `replicaCount` to 0:

   ```
   kubectl edit gw <gw-cr-name>
   ```

   For example:

   ```
   ...
   spec:
     replicaCount: 0
   ...
   ```

3. Wait for the gateway firmware pods to scale down and terminate. Ensure that the gateway firmware pods are terminated before you move to next step.
4. Scale up the gateway firmware containers back to their original value, or remove the `replicaCount` field if none was there before.

   ```
   kubectl edit gw <gw-cr-name>
   ```

5. Your operands should all have an upgraded version. To verify, wait for the cluster status to become `Running`, and Reconciled version to show the new version number.
   For example:

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

   ```
   NAME                                                    READY   STATUS    VERSION              RECONCILED VERSION
   AGE
   analyticscluster.analytics.apiconnect.example.com/analytics   8/8     Running   10.0.4.0   10.0.4.0-1074   121m

   NAME                                        PHASE     READY   SUMMARY                          VERSION    AGE
   datapowerservice.datapower.example.com/gw1   Running   True    StatefulSet replicas ready: 1/1   10.0.4.0   100m

   NAME                                        PHASE     LAST EVENT   WORK PENDING   WORK IN-PROGRESS   AGE
   datapowermonitor.datapower.example.com/gw1   Running                false          false              100m

   NAME                                           READY   STATUS    VERSION              RECONCILED VERSION      AGE
   gatewaycluster.gateway.apiconnect.example.com/gw1   2/2     Running   10.0.4.0   10.0.4.0-1074  100m

   NAME                                             READY   STATUS    VERSION              RECONCILED VERSION      AGE
   managementcluster.management.apiconnect.example.com/m1   16/16   Running   10.0.4.0   10.0.4.0-1074   162m


   NAME                                          READY   STATUS    VERSION              RECONCILED VERSION      AGE
   portalcluster.portal.apiconnect.example.com/portal   3/3     Running   10.0.4.0   10.0.4.0-1074   139m
   ```

   Troubleshooting: If you find your gateways do not sync with your management service, see known upgrade issues.

Upgrading on OpenShift and Cloud Pak for Integration

You can upgrade API Connect in an OpenShift environment and as part of IBM Cloud Pak for Integration.

- **Planning your API Connect upgrade on OpenShift**
  Review the supported upgrade paths and other upgrade considerations on OpenShift or Cloud Pak for Integration.
- **Pre-upgrade preparation and checks on OpenShift**
  Prepare your OpenShift or Cloud Pak for Integration API Connect deployment for upgrade.
- **Upgrading API Connect on OpenShift**
  Upgrade your API Connect installation to a newer version on OpenShift in an online (connected to the internet) environment.
- **Troubleshooting upgrade on OpenShift**
  Review the following troubleshooting tips if you encounter a problem during API Connect upgrade on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Planning your API Connect upgrade on OpenShift

Review the supported upgrade paths and other upgrade considerations on OpenShift or Cloud Pak for Integration.

Supported upgrade paths for API Connect

Table 1 lists the supported releases of IBM® API Connect v10 that can be upgraded directly to the new version of API Connect.

Table 1. Supported upgrade paths on OpenShift

Upgrade from:	How to upgrade to 10.0.7.0

Upgrade from:	How to upgrade to 10.0.7.0
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	Complete the procedure in Upgrading on OpenShift and Cloud Pak for Integration.
10.0.5.2 or earlier	Complete two upgrades: 1. Upgrade to version 10.0.5.3 or later as explained in Upgrading on OpenShift and Cloud Pak for Integration in the API Connect 10.0.5.x documentation. 2. Upgrade to the current release by completing the procedures in Upgrading on OpenShift and Cloud Pak for Integration.

Attention: Different releases of API Connect support different versions of operators, CASE files, and OpenShift. For more information, see Operator, CASE, and platform requirements for upgrading.

Upgrading API Connect in Cloud Pak for Integration

Cloud Pak for Integration 2023.4 uses API Connect 10.0.7.0. Before you can upgrade to API Connect 10.0.7.0, you must upgrade to API Connect 10.0.5.3 or later to ensure support for the newer versions of OpenShift and foundational services.

Supported DataPower Gateway versions

You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

However, before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. When you find the information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

Important upgrade considerations

Carefully review the following items, to plan and prepare for your upgrade.

Upgrading multiple instances of API Connect

- If a single operator manages multiple instances, all of those instances must be upgraded, one at a time, as soon as possible. The operator should not be managing an operand based on an older version any longer than necessary.
- Ensure that each instance is fully upgraded before you start the upgrade on the next instance.

API transaction failure during the upgrade of gateway

The gateway subsystem remains available and continues processing API calls during the upgrade of the management, portal, and analytics subsystems. However, a few API transactions might fail during the upgrade of the gateway.

When you upgrade a cluster of gateway pods, a few API transactions might fail. During the upgrade, Kubernetes removes the pod from the load balancer configuration, deletes the pod, and then starts a new pod. The steps are repeated for each pod. Socket hang-ups occur on transactions that are in process at the time the pod is terminated.

The number of transactions that fail depends on the rate of incoming transactions, and the length of time that is needed to complete each transaction. Typically the number of failures is a small percentage. This behavior is expected during an upgrade. If the failure level is not acceptable, schedule the upgrade during an off-hours maintenance window.

DataPower Gateway supports long-lived connections such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when you upgrade. Workloads with long-lived connections are more vulnerable to failed API transactions during upgrading.

You can limit the number of failed API transactions during the upgrade by using the DataPower Operator's `lifecycle` property to configure the `preStop` container lifecycle hook on the gateway pods. This approach mitigates the risk of API failures during the rolling update of the gateway StatefulSet by sleeping the pod for a span of time, allowing in-flight transactions to complete before the SIGTERM is delivered to the container. While this feature does not guarantee no in-flight API call failures, it does provide some mitigation for in-flight transactions that can complete successfully within the configured time window. For more information, see Delaying SIGTERM with preStop in the DataPower documentation.

OAuth provider resources with `api.securityDefintions`

If you have native OAuth providers that are configured with the `api.securityDefintions` field assigned, then upgrade fails. Before upgrade, remove all `api.securityDefintions` from all native OAuth providers that you configured in the Cloud Manager and API Manager UIs:

- Configuring a native OAuth provider in the Cloud Manager UI
- Configuring a native OAuth provider in the API Manager UI

Switch to the source view and delete any `securityDefintions` sections that are present.

SFTP management backups (V10.0.6.0 to V10.0.7.0 upgrade)

API Connect V10.0.7.0 supports only backups to an S3 object-store for the management database. If your V10.0.6.0 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.

Upgrade to V10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store.

If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block. Support for SFTP and local backups will be provided in a future V10.0.x release.

Extra backup folder appended to management database backup path

When you upgrade to V10.0.7.0 or later with S3 backups configured, an extra backup path folder /edb is appended to your backup path. The /edb path is to distinguish new EDB backups from your previous management backups.

Extra persistent storage requirement for management subsystem
Upgrade to V10.0.7.0 or later requires extra persistent storage space for the management subsystem. Check that your environment has at least as much extra capacity as the capacity assigned to your existing Postgres PVC:

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

```
NAME                                     STATUS   VOLUME              CAPACITY   ACCESS MODES   STORAGECLASS    AGE
def-management-site1-postgres            Bound    local-pv-7b528204   244Gi      RWO            local-storage   125d
def-management-site1-postgres-pgbr-repo  Bound    local-pv-8d425510   244Gi      RWO            local-storage   125d
def-management-site1-postgres-wal        Bound    local-pv-ee8feb38   244Gi      RWO            local-storage   125d
```

Analytics persistent queue enablement requires extra PV. Upgrades from V10.0.6 and V10.0.5
The analytics persistent queue feature is enabled during the upgrade (if it was not already enabled). This feature requires an extra Physical Volume (PV). Before you upgrade from V10.0.6 or V10.0.5.x, ensure that your environment has an available PV.

- **Operator, CASE, and platform requirements for upgrading**
  Before you upgrade API Connect on OpenShift Container Platform, verify that you are using the correct operator, CASE file (for air-gapped upgrades), and OpenShift.

Operator, CASE, and platform requirements for upgrading

Before you upgrade API Connect on OpenShift Container Platform, verify that you are using the correct operator, CASE file (for air-gapped upgrades), and OpenShift.

Supported versions of operators for API Connect

Table 1 lists the operators that are used for installing and upgrading API Connect.

Table 1. Operators for installing and upgrading API Connect

API Connect version	Data Power version	API Connect Operator channel	API Connect Operator version	DataPower Operator version	Foundational services channel version
10.0.7.0	10.5.3.0	5.1	5.1.0	1.9.0	4.3 to 4.6
10.0.6.0	10.5.1.0	5.0	5.0.0	1.7.0	3.23

CASE versions required for air-gapped installation and upgrade

Table 2 lists the required CASE file for each API Connect release:

Table 2. CASE versions for air-gapped deployments

API Connect version	API Connect CASE version	Foundational services CASE version	EDB CASE version
10.0.7.0	5.0.1	5.0.0	4.18.0
10.0.6.0	5.0.0	1.15.10	(none)

Supported versions of Red Hat OpenShift Container Platform

Table 3 provides a quick reference of the supported OpenShift versions for each API Connect release:

Table 3. API Connect and OpenShift Container Platform (OCP) compatibility

API Connect version	OCP 4.10	OCP 4.12	OCP 4.14
10.0.5.3	✓	✓	
10.0.5.4	✓	✓	
10.0.5.5	✓	✓	✓
10.0.6.0	✓	✓	
10.0.7.0		✓	✓

Upgrading OpenShift to support a newer version of API Connect

Attention: To avoid compatibility issues, do not upgrade your current version of OpenShift until after the API Connect upgrade is complete.
When you upgrade API Connect, both your current deployment and your target release must support the same version of OpenShift. After the API Connect upgrade, you can optionally update OpenShift to the highest version supported by the new release of API Connect.

During the upgrade process, it is possible that API Connect is temporarily running on an unsupported version of OpenShift because you must upgrade API Connect before you upgrade OpenShift. For example, suppose that your current deployment is running API Connect 10.0.5.4 on OpenShift 4.10 and you want to upgrade to API Connect 10.0.7.0 on OpenShift 4.14. The highest version of OpenShift that API Connect 10.0.5.4 supports is 4.12, but you can upgrade to OpenShift 4.14 after you complete the API Connect upgrade to 10.0.7.0.

In this scenario, the upgrade path is as follows:

1. On your API Connect 10.0.5.4 deployment, upgrade API Connect to 10.0.7.0.
2. On the API Connect 10.0.7.0 deployment, upgrade OpenShift to versions 4.11 and 4.12 to reach the minimum supported version (all interim upgrades are required on OpenShift).
3. Upgrade OpenShift from 4.12 to 4.13 and then 4.14 to reach the latest supported version of OpenShift.

For OpenShift upgrade instructions, see the Red Hat OpenShift documentation.

Pre-upgrade preparation and checks on OpenShift

Prepare your OpenShift or Cloud Pak for Integration API Connect deployment for upgrade.

This topic covers all the preparation steps you must complete before you can start the upgrade process.

Before following the steps in this topic, review the Planning your API Connect upgrade on OpenShift to ensure that you are following a supported upgrade path and that you understand important changes that might affect your deployment.

The Pre-upgrade health-checks that are documented in this topic should always be run just before you start the upgrade process, but you can also run them at any time.
Tip: Run the Pre-upgrade health-checks during your upgrade planning stage to identify any problems as early as possible. Run the checks again just before upgrade to confirm API Connect is still ready for upgrade.

Pre-upgrade health-checks

1. Check that the API Connect operator and subsystems are all running.
   To verify the health of each subsystem, run the following commands:

   ```
   oc get ManagementCluster -n <management namespace>
   oc get GatewayCluster -n <gateway namespace>
   oc get PortalCluster -n <portal namespace>
   oc get AnalyticsCluster -n <analyticsnamespace>
   ```

   Confirm that all pods are **READY**, for example:

   ```
   oc get PortalCluster -n apic
   NAME      READY   STATUS    VERSION    RECONCILED VERSION   AGE
   portal    3/3     Running   10.0.7.0   10.0.7.0-95    57m
   ```

2. Verify that the management subsystem's **pgcluster** is working correctly:
   a. Get the name of the **pgcluster**:

      ```
      oc get pgcluster -n <management namespace>
      ```

      The response displays the name of the postgres cluster that is running in the specified namespace.

   b. Check the status of the **pgcluster**:

      ```
      oc get pgcluster <pgcluster_name> -n <APIC_namespace> -o yaml | grep status -A 2 | tail -n3
      ```

      A successful response looks like the following example, where the state is **Initialized**:

      ```
      status:
          message: Cluster has been initialized
          state: pgcluster Initialized
      ```

      Important: If the **pgcluster** returns any other state, then do not proceed with the upgrade.
      - If backup or restore jobs are running, wait until they complete and then check the status again. Do not proceed with the upgrade until the status is **Initialized**.
      - If all background jobs are complete but the **pgcluster** does not return **state: pgcluster Initialized**, then contact IBM Support for assistance.
3. Run the pre-upgrade health check:
   Attention: The pre-upgrade health check is a required step. Failure to run this check might result in upgrade failure that requires disaster recovery to restore your API Connect deployment.
   Note: If you have a 2DCDR deployment, run this check only on the active management subsystem.
   a. Verify that the **apicops** utility is installed and that you have the latest version by running the following command:

      ```
      apicops --version
      ```

      For more information about the **apicops** utility, see The API Connect operations tool: apicops.

   b. Run the following command to set the KUBECONFIG environment variable.

      ```
      export KUBECONFIG=</path/to/kubeconfig>
      ```

   c. Run the **apicops pre-upgrade** check:

      ```
      apicops version:pre-upgrade -n <namespace>
      ```

      If your API Connect deployment is working correctly, the output shows no error messages.

      Note: The **apicops pre-upgrade** command asks for the Cloud Manager admin password so that it can use the platform API.
      If you do not have this password, or the platform API URL is not accessible from your **apicops** location, then this part of the pre-upgrade check can be skipped by adding the argument **--no-topology**.

      The topology check output that the API call produces is typically only required for debugging purposes, when the **apicops version:pre-upgrade** script identifies problems with your installation.

   Note: Cloud Pak for Integration: If you are upgrading API Connect in Cloud Pak for Integration and the pre-upgrade check indicates duplicate user accounts in the registry, resolve them now as explained in Resolving duplicate users before upgrade on Cloud Pak for Integration.

Additional pre-upgrade operations

Extra operations to review and complete before you start the upgrade.

1. If you are upgrading to V10.0.7.0, and you do not have management database backups configured, or you have them configured for SFTP or local, then change your management database backup settings to S3. See Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

   V10.0.7.0 supports only backups to an S3 object-store. If you are not able to configure an S3 object-store, then to upgrade to v10.0.7.0 you must delete the **databaseBackup** section from your management CR, and add the **apiconnect-operator/backups-not-configured: "true"** annotation before you upgrade.

2. If you are upgrading to V10.0.7.0, and the analytics persistent queue is not enabled, check that you have an extra Physical Volume available.
   For more information on this requirement, see Analytics persistent queue enablement requires extra PV.

3. If you used any microservice image overrides in any of your API Connect subsystem CRs, remove the image overrides before upgrade.
4. If you are upgrading an air-gapped (disconnected from the internet) installation, configure the catalog sources for the target API Connect version on your bastion host or portable storage device:
   - Pre-upgrade preparation for air-gapped environment using a bastion host.
   - Pre-upgrade preparation for air-gapped environment using a portable storage device.

Backup your API Connect deployment

If upgrade fails, rollback is not possible. Before upgrade complete the disaster recovery preparation for all your subsystems.

For instructions on backing up and disaster recovery preparation, see the following topics:

- v10.0.7.0 and later: Backing up, restoring, and disaster recovery.
- v10.0.6.0: Backups on OpenShift V10.0.6.0, Disaster recovery on OpenShift and Cloud Pak for Integration v10.0.6.0.
- v10.0.5.x: Backups on OpenShift and Cloud Pak for Integration, Disaster recovery on OpenShift and Cloud Pak for Integration.

- **Resolving duplicate users before upgrade on Cloud Pak for Integration**
  If you are upgrading API Connect in Cloud Pak for Integration and the API Connect pre-upgrade check indicates duplicate user accounts in the registry, resolve them before you proceed with the upgrade.
- **Pre-upgrade preparation for air-gapped environment using a bastion host**
  In a disconnected, network-restricted environment, prepare your API Connect upgrade sources on your bastion host.
- **Pre-upgrade preparation for air-gapped environment using a portable storage device**
  In a disconnected, network-restricted environment, prepare your API Connect upgrade sources on your portable storage device.

Resolving duplicate users before upgrade on Cloud Pak for Integration

If you are upgrading API Connect in Cloud Pak for Integration and the API Connect pre-upgrade check indicates duplicate user accounts in the registry, resolve them before you proceed with the upgrade.

Before you begin

This task applies only when you are upgrading API Connect in Cloud Pak for Integration from 2023.2 (or earlier) to 2023.4 (or later). Be sure to run the pre-upgrade check as explained in Pre-upgrade preparation and checks on OpenShift to determine whether duplicate user records exist in your deployment.

About this task

Starting with 2023.4 (API Connect 10.0.7.0), Cloud Pak for Integration uses Keycloak as an OIDC provider to authenticate users instead of IAM (Identity and Access Management). Due to differences in how Keycloak and IAM treat usernames, you might need to manually delete duplicate user accounts to ensure that users can log in after the upgrade. Review the following scenarios to determine whether you need to manually delete any user accounts.

In a user registry, each user is uniquely identified by their username. If a registry allows for case-sensitive usernames, then two user records with matching usernames that use different cases (for example, **Alice** and **ALICE**) can exist and they point to two different users. If the registry is case-insensitive, then **Alice** and **ALICE** point to the same user.

If the user registry that is backing IAM supports case-sensitive usernames, then IAM uses the case of the username from that registry. For example, if IAM is backed by an LDAP directory that contains **ALICE** as the username, then IAM also uses **ALICE** as the username.

Keycloak supports only lowercase letters for usernames, so **ALICE** from the LDAP directory is stored with the **preferred_name alice** in Keycloak.

Assuming **ALICE** is the username in the LDAP directory and was used by IAM before the API Connect upgrade, then API Connect might contain either **alice** or **ALICE** in its database after the upgrade, depending on how that user logged in to API Connect after the upgrade:

- If the user logged in using the Automation UI in Cloud Pak for Integration, then API Connect created a lowercase username (**alice**).
- If the user logged in using the API Connect toolkit, or an invitation from API Connect, then API Connect created a username that matches the case that was used in the underlying registry (**ALICE**).

It is possible that two user records now exist for the same user because one might be in lowercase letters and the other might be in uppercase or mixed-case letters. To determine whether duplicate user records exist and then resolve them, complete the following steps.

Procedure

1. Review the results of the pre-upgrade check to determine whether your deployment includes duplicate users.
   If duplicate user records are detected, the script returns a response similar to the following example (note the duplicated **username** setting):

```
** Checking for duplicate users in the common-services user registry **
Command: apicops-v10 duplicate-users
*****************************
```

```
getDuplicateUsers::getDuplicateUsers Entering
Found duplicate user entries: [
    {
        "username": "alice",
        "id": "291339db-acd9-493d-98c2-cfd47185588e",
        "name": "alice",
        "title": "alice",
        "first_name": "ALICE",
        "last_name": "CAPS",
        "email": "alice@gmail.com",
        "identity_provider": "common-services",
        "created_at": "2023-11-22T21:43:51.000Z",
        "updated_at": "2023-11-28T19:17:42.000Z",
        "last_login_at": "2023-11-28T19:17:42.000Z",
        "url": "/api/user-registries/118b7e20-3f97-40ce-bc00-936f202d52a4/6f69a530-4192-4610-ba1f-
15ea9711284e/users/291339db-acd9-493d-98c2-cfd47185588e"
    },
    {
        "username": "ALICE",
        "id": "6325084d-91a0-4c1e-9743-d615105ce4e3",
        "name": "ALICE-from-common-services-of-type-standard",
        "title": "ALICE from common-services of type standard",
        "first_name": "ALICE",
        "last_name": "ALICE",
        "email": "alice@gmail.com",
        "identity_provider": "common-services",
        "created_at": "2023-11-22T21:59:41.000Z",
        "updated_at": "2023-11-22T21:59:48.000Z",
        "last_login_at": "2023-11-22T21:59:48.000Z",
        "url": "/api/user-registries/118b7e20-3f97-40ce-bc00-936f202d52a4/6f69a530-4192-4610-ba1f-
15ea9711284e/users/6325084d-91a0-4c1e-9743-d615105ce4e3"
    }
]
================================================
 WARNING:
================================================
 Duplicate user records have been detected in the common-services user registry.
 Duplicate records will NOT be fixed by the system and might cause unexpected behavior.
 Please refer to product documentation on steps to clean up the duplicate entries.
================================================
****************************

***********************************************
** Finished: apicops-v10 version:pre-upgrade **
***********************************************
```

In this example, duplicate records were detected, with the username "alice" and "ALICE".

If your results do not indicate any duplicate users, skip the rest of this task. If the results indicate any duplicates, continue with the remaining steps to resolve the user records.

2. Decide which user account you want to preserve; the other account is considered to be the duplicate.
   You can retain either account. The choice is a matter of convenience. For example, if one of the accounts functions as the owner of organizations, you might find it simpler to retain that account to avoid having to transfer ownership to the other account. You do not have to rename the account because you delete the duplicate account; the username of the preserved account is converted to lowercase during the upgrade. For example, if you choose to preserve user "ALICE" and you delete the record for "alice" then the **username** for "ALICE" will be converted to "alice" the first time the user logs in after the upgrade.

3. If needed, transfer ownership of assets to the account that you want to preserve:
   a. Transfer ownership of catalogs.
      For instructions, see Transfer Catalog Owner on GitHub.

   b. Transfer ownership of spaces within catalogs.
      For instructions, see Transfer Space Owner on GitHub.

   c. Transfer ownership of a provider organization.
      For instructions, see Transfer Org Owner on GitHub.

4. Add any missing roles to the account that you want to preserve.
   Check the roles that are assigned to the duplicate account. If the account that you want to preserve does not include those roles, assign them now. For instructions, see Assigning roles to members in this documentation.

5. Delete the duplicate user account.
   For instructions, see Removing a user from a user registry in this documentation.

Pre-upgrade preparation for air-gapped environment using a bastion host

In a disconnected, network-restricted environment, prepare your API Connect upgrade sources on your bastion host.

Before you begin

- Review and complete all steps that are documented in Planning your API Connect upgrade on OpenShift and Pre-upgrade preparation and checks on OpenShift.
- The upgrade procedure requires you to use Red Hat Skopeo for moving container images. Skopeo is not available for Microsoft Windows, so you cannot do this task with a Windows host.

Attention: Some steps in this task use tilde character: ~. Do not use this character within double quotation marks in any command because the tilde doesn't expand and the command might fail.

About this task

If you previously deployed or upgraded API Connect in an air-gapped environment, you can use the same bastion host as before.

Procedure

1. Set up the mirroring environment.
   a. Prepare the target cluster:
      - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
        For information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in IBM API Connect Version 10 software product compatibility requirements.
      - Configure storage on the cluster and make sure that it is available.

   b. Prepare the bastion host:
      You must be able to connect your bastion host to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry) at the same time. Your host must be on a Linux x86_64 or Mac platform with any operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

      i. Ensure that the sites and ports listed in Table 1 can be reached from the bastion host:

      Table 1. Sites that must be reached from the bastion host

Site	Description
`icr.io:443`	IBM entitled registry
`quay.io:443`	Local API Connect image repository
`github.com`	CASE files and tools
`redhat.com`	Red Hat OpenShift Container Platform (OCP) upgrades

      ii. On the bastion host, install either Docker or Podman (not both).
      Docker and Podman are used for managing containers; you only need to install one of these applications.

      - To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

        ```
        yum check-update
        yum install docker
        ```

      - To install Podman, see the Podman installation instructions.
        For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

        ```
        yum install podman
        ```

      iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in Getting started with the OpenShift CLI.
      The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

      iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
      The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

      ```
      oc ibm-pak --help
      ```

   c. Set up a local image registry and credentials.
      The local Docker registry stores the mirrored images in your network-restricted environment.

      i. Install a registry, or get access to an existing registry.
      You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

      The registry product that you use must meet the following requirements:
      - Supports multi-architecture images through Docker Manifest V2, Schema 2
        For details, see Docker Manifest V2, Schema 2.
      - Is accessible from the Red Hat OpenShift Container Platform cluster nodes
      - Allows path separators in the image name

      Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

      ii. Configure the registry to meet the following requirements:
      - Supports auto-repository creation
      - Has sufficient storage to hold all of the software that is to be transferred
      - Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials)
      - Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials)

      To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries during runtime, use an account that can read from the target local registry.

2. Mirror the cert-manager operator for Red Hat OpenShift.
   Mirror the operator as explained in the Red Hat instructions: Mirroring images for a disconnected installation using the oc-mirror plugin .

   When you modify the `ImageSetConfiguration`, include `openshift-cert-manager-operator` at channel `stable-v1` as one of the packages, as shown in the following example:

   ```
   packages:
   - name: openshift-cert-manager-operator
     channels:
     - name: stable-v1
   ```

3. Set environment variables and download the CASE files.
   Create environment variables to use while mirroring images, connect to the internet, and download the CASE files needed for installing API Connect.

a. Create the following environment variables with the installer image name and the image inventory on your host:
- API Connect:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64
```

For information about API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

- Foundational services:

```
export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=1.15.10
export CS_ARCH=amd64
```

For information about IBM Cloud Pak foundational services (common services) CASE versions, see "Table 1. Image versions for offline installation" in Installing IBM Cloud Pak foundational services in an air-gapped environment in the IBM Cloud Pak foundational services documentation.

- EDB:

```
export EDB_CASE_NAME=ibm-cloud-native-postgresql
export EDB_CASE_VERSION=4.18.0
```

b. Connect your host to the internet (it does not need to be connected to the network-restricted environment yet).
c. Download the CASE files to your host:
- API Connect:

```
oc ibm-pak get $CASE_NAME --version $CASE_VERSION
```

- Foundational services:

```
oc ibm-pak get $CS_CASE_NAME --version $CS_CASE_VERSION
```

- EDB:

```
oc ibm-pak get $EDB_CASE_NAME --version $EDB_CASE_VERSION
```

4. Generate mirror manifests.
The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before you install API Connect.

a. Define the environment variable $TARGET_REGISTRY by running the following command:

```
export TARGET_REGISTRY=<target-registry>
```

Replace **<target-registry>** with the IP address (or hostname) and port of the local registry; for example: **172.16.0.10:5000**. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: **172.16.0.10:5000/registry_ns**.
b. Generate mirror manifests by running the following commands:
- API Connect:

```
oc ibm-pak generate mirror-manifests $CASE_NAME --version $CASE_VERSION $TARGET_REGISTRY
```

- Foundational services:

```
oc ibm-pak generate mirror-manifests $CS_CASE_NAME --version $CS_CASE_VERSION $TARGET_REGISTRY
```

- EDB:

```
oc ibm-pak generate mirror-manifests $EDB_CASE_NAME --version $EDB_CASE_VERSION $TARGET_REGISTRY
```

If you need to filter for a specific image group, add the parameter **--filter** **<image_group>** to the command.

The **generate** command creates the following files at ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION, ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION, and ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION:
- catalog-sources.yaml
- catalog-sources-linux-*<arch>*.yaml (if there are architecture-specific catalog sources)
- image-content-source-policy.yaml
- images-mapping.txt

The files are used when mirroring the images to the **TARGET_REGISTRY**.

5. Authenticate with the entitled registry where the images are hosted.
a. Obtain an entitlement key for the registry where the images are hosted:
i. Log in to the IBM Container Library.
ii. In the Container software library, select Get entitlement key.
iii. In the "Access your container software" section, click Copy key.
iv. Copy the key to a safe location; you will use it to log in to **cp.icr.io** in the next step.
b. Login to **cp.icr.io** registry:
The image pull secret is used to authenticate with the registry and access product images.

i. Run the following command to export the path to the file that stores the authentication credentials that are generated on a Podman or Docker login:

```
export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
```

The authentication file is typically located in $HOME/.docker/config.json on Linux or %USERPROFILE%/.docker/config.json on Windows.

ii. Log in to the **cp.icr.io** registry with Podman or Docker; for example:

```
podman login cp.icr.io
```

Use **cp** as the username and your entitlement key as the password.

6. Authenticate with the local registry.
   Log in to the local registry using an account that can write images to that registry; for example:

   ```
   podman login $TARGET_REGISTRY
   ```

   If the registry is insecure, add the following flag to the command: **--tls-verify=false**.

7. Mirror the product images.
   a. Connect the bastion host to both the internet and the restricted-network environment that contains the local registry.
   b. Run the following commands to copy the images to the local registry:
      - API Connect:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1
        ```

      - Foundational services:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1
        ```

      - EDB:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1
        ```

     Note: If the local registry does not use TLS, or your device does not trust the certificate that the local registry presents, add the **--insecure** option to the command.
     There might be a slight delay before you see a response to the command.

8. Configure the target cluster.
   Now that images are mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

   a. Log in to your Red Hat OpenShift Container Platform cluster:

      ```
      oc login <openshift_url> -u <username> -p <password> -n <namespace>
      ```

   b. Update the global image pull secret for the cluster as explained in the Red Hat OpenShift Container Platform documentation.
      Updating the image pull secret provides the cluster with the credentials that are needed for pulling images from your local registry.

      Note: If you have an insecure registry, add the registry to the cluster's **insecureRegistries** list by running the following command:

      ```
      oc edit image.config.openshift.io/cluster -o yaml
      ```

      and add the TARGET_REGISTRY to **spec.registrySources.insecureRegistries** as shown in the following example:

      ```
      spec:
        registrySources:
          insecureRegistries:
          - insecure0.svc:5001
          - <TARGET_REGISTRY>
      ```

      If the **insecureRegistries** field does not exist, you can add it.
   c. Create the ImageContentSourcePolicy, which instructs the cluster to pull the images from your local registry:
      - API Connect:

        ```
        oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
        ```

      - Foundational services:

        ```
        oc apply -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/image-content-source-policy.yaml
        ```

      - EDB:

        ```
        oc apply -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/image-content-source-policy.yaml
        ```

   d. Verify that each ImageContentSourcePolicy resource was created:

      ```
      oc get imageContentSourcePolicy
      ```

   e. Verify your cluster node status:

      ```
      oc get MachineConfigPool -w
      ```

      Wait for all nodes to be updated before you proceed to the next step.

9. Apply the catalog sources.

Now that the images are mirrored to the target cluster, apply the catalog sources for API Connect and Foundational services. You do not need to apply the catalog source for EDB because the API Connect operator applies that catalog source in its own namespace when the API Connect operand is created.

In the following steps, replace **<*Architecture*>** with either **amd64**, **s390x** or **ppc64le** according to your environment.

a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64

export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=5.0.0
export CS_ARCH=amd64
```

b. Generate the catalog source and save it in another directory in case you need to replicate this installation in the future.

i. Get the catalog source:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

cat ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

c. Apply the catalog sources to the cluster.

i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

oc apply -f ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

ii. Apply any architecture-specific catalog source:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

iii. Confirm that the catalog source was created in the **openshift-marketplace** namespace:

```
oc get catalogsource -n openshift-marketplace
```

What to do next

Upgrade your API Connect deployment, following the procedure documented in Upgrading API Connect on OpenShift.

Pre-upgrade preparation for air-gapped environment using a portable storage device

In a disconnected, network-restricted environment, prepare your API Connect upgrade sources on your portable storage device.

Before you begin

- Review and complete all steps that are documented in Planning your API Connect upgrade on OpenShift and Pre-upgrade preparation and checks on OpenShift.
- The upgrade procedure requires you to use Red Hat Skopeo for moving container images. Skopeo is not available for Microsoft Windows, so you cannot do this task with a Windows host.

Attention: Some steps in this task use tilde character: ~. Do not use this character within double quotation marks in any command because the tilde doesn't expand and the command might fail.

About this task

If you previously deployed or upgraded API Connect in an air-gapped environment, you can use the same portable storage device as before.

Procedure

1. Set up the mirroring environment.
   a. Prepare the target cluster:
      - Deploy a supported version of Red Hat OpenShift Container Platform (OCP) as a cluster.
        For more information, see Table 2 "API Connect and OpenShift Container Platform (OCP) compatibility matrix" in Operator, CASE, and platform requirements.
      - Configure storage on the cluster and make sure that it is available.

   b. Prepare the portable device:
      You must be able to connect your portable device to the internet and to the restricted network environment (with access to the Red Hat OpenShift Container Platform (OCP) cluster and the local registry). The portable device must be on a Linux x86_64 or Mac platform with an operating system that the Red Hat OpenShift Client supports (in Windows, execute the actions in a Linux x86_64 VM or from a Windows Subsystem for Linux terminal).

      i. Check that the portable device has sufficient storage to hold all the software that is to be transferred to the local registry.
      ii. On the portable device, install either Docker or Podman (not both).
      Docker and Podman are used for managing containers; you can install either of these applications.

- To install Docker (for example, on Red Hat Enterprise Linux), run the following commands:

  ```
  yum check-update
  yum install docker
  ```

- To install Podman, see the Podman installation instructions.
  For example, on Red Hat Enterprise Linux 9, install Podman with the following command:

  ```
  yum install podman
  ```

iii. Install the Red Hat OpenShift Client tool (`oc`) as explained in Getting started with the OpenShift CLI.
The `oc` tool is used for managing Red Hat OpenShift resources in the cluster.

iv. Download the IBM Catalog Management Plug-in for IBM Cloud Paks version 1.1.0 or later from GitHub.
The `ibm-pak` plug-in enables you to access hosted product images, and to run `oc ibm-pak` commands against the cluster. To confirm that `ibm-pak` is installed, run the following command and verify that the response lists the command usage:

```
oc ibm-pak --help
```

c. Set up a local image registry and credentials.
The local Docker registry stores the mirrored images in your network-restricted environment.

i. Install a registry, or get access to an existing registry.
You might already have access to one or more centralized, corporate registry servers to store the API Connect images. If not, then you must install and configure a production-grade registry before proceeding.

The registry product that you use must meet the following requirements:
- Supports multi-architecture images through Docker Manifest V2, Schema 2.
  For more information, see Docker Manifest V2, Schema 2.
- Is accessible from the Red Hat OpenShift Container Platform cluster nodes.
- Allows path separators in the image name.

Note: Do not use the Red Hat OpenShift image registry as your local registry because it does not support multi-architecture images or path separators in the image name.

ii. Configure the registry to meet the following requirements:
- Supports auto-repository creation.
- Has sufficient storage to hold all the software that is to be transferred.
- Has the credentials of a user who can create and write to repositories (the mirroring process uses these credentials).
- Has the credentials of a user who can read all repositories (the Red Hat OpenShift Container Platform cluster uses these credentials).

To access your registries during an air-gapped installation, use an account that can write to the target local registry. To access your registries at run time, use an account that can read from the target local registry.

2. Mirror the cert-manager operator for Red Hat OpenShift.
Mirror the operator as explained in the Red Hat instructions: Mirroring images for a disconnected installation using the oc-mirror plugin .

When you modify the `ImageSetConfiguration`, include `openshift-cert-manager-operator` at channel `stable-v1` as one of the packages, as shown in the following example:

```
  packages:
  - name: openshift-cert-manager-operator
    channels:
    - name: stable-v1
```

3. Set environment variables and download the CASE files.
Create environment variables to use while mirroring images, connect to the internet, and download the CASE files needed for installing API Connect.

a. Create the following environment variables with the installer image name and the image inventory on your portable device:
- API Connect:

  ```
  export CASE_NAME=ibm-apiconnect
  export CASE_VERSION=5.1.0
  export ARCH=amd64
  ```

  For information about API Connect CASE versions and their corresponding operators and operands, see Operator, CASE, and platform requirements.

- Foundational services:

  ```
  export CS_CASE_NAME=ibm-cp-common-services
  export CS_CASE_VERSION=1.15.10
  export CS_ARCH=amd64
  ```

  For information about IBM Cloud Pak foundational services (common services) CASE versions, see "Table 1. Image versions for offline installation" in Installing IBM Cloud Pak foundational services in an air-gapped environment in the IBM Cloud Pak foundational services documentation.

- EDB:

  ```
  export EDB_CASE_NAME=ibm-cloud-native-postgresql
  export EDB_CASE_VERSION=4.18.0
  ```

b. Connect your portable device to the internet (it does not need to be connected to the network-restricted environment yet).
c. Download the CASE files to your portable device:
- API Connect:

  ```
  oc ibm-pak get $CASE_NAME --version $CASE_VERSION
  ```

- Foundational services:

  ```
  oc ibm-pak get $CS_CASE_NAME --version $CS_CASE_VERSION
  ```

- EDB:

```
oc ibm-pak get $EDB_CASE_NAME --version $EDB_CASE_VERSION
```

4. Generate mirror manifests.
   The process of mirroring images pulls the images from the internet and pushes them to your local registry. After mirroring your images, you can configure your cluster and pull the images to it before you install API Connect.

   a. Define the environment variable $TARGET_REGISTRY by running the following command:

   ```
   export TARGET_REGISTRY=<target-registry>
   ```

   Replace **<target-registry>** with the IP address (or hostname) and port of the local registry; for example: **172.16.0.10:5000**. If you want the images to use a specific namespace within the target registry, you can specify it here; for example: **172.16.0.10:5000/registry_ns**.

   b. Generate mirror manifests by running the following commands:

   - API Connect:

     ```
     oc ibm-pak generate mirror-manifests $CASE_NAME --version $CASE_VERSION $TARGET_REGISTRY
     ```

   - Foundational services:

     ```
     oc ibm-pak generate mirror-manifests $CS_CASE_NAME --version $CS_CASE_VERSION $TARGET_REGISTRY
     ```

   - EDB:

     ```
     oc ibm-pak generate mirror-manifests $EDB_CASE_NAME --version $EDB_CASE_VERSION $TARGET_REGISTRY
     ```

   If you need to filter for a specific image group, add the parameter **--filter** **<image_group>** to the command.

   The **generate** command creates the following files at ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION, ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION, and ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION:

   - catalog-sources.yaml
   - catalog-sources-linux-<*arch*>.yaml (if there are architecture-specific catalog sources)
   - image-content-source-policy.yaml
   - images-mapping-to-filesystem.txt
   - images-mapping-from-filesystem.txt

   The files are used when mirroring the images to the **TARGET_REGISTRY**.

5. Authenticate with the entitled registry where the images are hosted.

   a. Obtain an entitlement key for the registry where the images are hosted:

      i. Log in to the IBM Container Library.
      ii. In the Container software library, select Get entitlement key.
      iii. In the "Access your container software" section, click Copy key.
      iv. Copy the key to a safe location; you will use it to log in to **cp.icr.io** in the next step.

   b. Login to **cp.icr.io** registry:
   The image pull secret is used to authenticate with the registry and access product images.

      i. Run the following command to export the path to the file that stores the authentication credentials that are generated on a Podman or Docker login:

      ```
      export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
      ```

      The authentication file is typically located in $HOME/.docker/config.json on Linux or %USERPROFILE%/.docker/config.json on Windows.

      ii. Log in to the **cp.icr.io** registry with Podman or Docker; for example:

      ```
      podman login cp.icr.io
      ```

      Use **cp** as the username and your entitlement key as the password.

6. Mirror the images from the internet to the portable device.

   a. Define the environment variable $IMAGE_PATH by running the following command:

   ```
   export IMAGE_PATH=<image-path>
   ```

   where **<image-path>** indicates where the files are stored on the portable device's file system.

   b. Mirror the images to the portable device:

   - API Connect:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-to-filesystem.txt \
       --filter-by-os '.*' \
       -a $REGISTRY_AUTH_FILE \
       --skip-multiple-scopes \
       --max-per-registry=1 \
       --dir "$IMAGE_PATH"
     ```

   - Foundational services:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping-to-filesystem.txt \
       --filter-by-os '.*' \
       -a $REGISTRY_AUTH_FILE \
       --skip-multiple-scopes \
       --max-per-registry=1 \
       --dir "$IMAGE_PATH"
     ```

   - EDB:

     ```
     oc image mirror \
       -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping-to-filesystem.txt \
       --filter-by-os '.*' \
       -a $REGISTRY_AUTH_FILE \
     ```

```
  --skip-multiple-scopes \
  --max-per-registry=1 \
  --dir "$IMAGE_PATH"
```

There might be a slight delay before you see a response to the command.

7. Move the portable device to the restricted-network environment.
   The procedure depends on the type of device that you are using:
   If you are using a portable computer, disconnect the device from the internet and connect it to the restricted-network environment. The same environment variables can be used.

   If you are using portable storage, complete the following steps:

   a. Transfer the following files to a device in the restricted-network environment:
      - The `~/.ibm-pak` directory.
      - The contents of the ***<image-path>*** that you specified in the previous step.
   b. Create the same environment variables as on the original device; for example:
      - API Connect:

        ```
        export CASE_NAME=ibm-apiconnect
        export CASE_VERSION=5.1.0
        export ARCH=amd64
        ```

      - Foundational services:

        ```
        export CS_CASE_NAME=ibm-cp-common-services
        export CS_CASE_VERSION=1.15.10
        export CS_ARCH=amd64
        ```

      - EDB:

        ```
        export EDB_CASE_NAME=ibm-cp-common-services
        export EDB_CASE_VERSION=1.15.10
        export EDB_ARCH=amd64
        ```

      - Registry:

        ```
        export REGISTRY_AUTH_FILE=$HOME/.docker/config.json
        ```

      - Image path:

        ```
        export IMAGE_PATH=<image-path>
        ```

8. Authenticate with the local registry.
   Log in to the local registry using an account that can write images to that registry; for example:

   ```
   podman login $TARGET_REGISTRY
   ```

   If the registry is insecure, add the following flag to the command: `--tls-verify=false`.

9. Mirror the product images to the target registry.
   a. Mirror the product images to the target registry.
      i. If you are using a portable computer, connect it to the restricted-network environment that contains the local registry.
      If you are using portable storage, you already transferred files to a device within the restricted-network environment.

      ii. Run the following commands to copy the images to the local registry:
      - API Connect:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/images-mapping-from-filesystem.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1 \
          --from-dir "$IMAGE_PATH"
        ```

      - Foundational services:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/images-mapping-from-filesystem.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1 \
          --from-dir "$IMAGE_PATH"
        ```

      - EDB:

        ```
        oc image mirror \
          -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/images-mapping-from-filesystem.txt \
          -a $REGISTRY_AUTH_FILE \
          --filter-by-os '.*' \
          --skip-multiple-scopes \
          --max-per-registry=1 \
          --from-dir "$IMAGE_PATH"
        ```

      Note: If the local registry does not use TLS, or your device does not trust the certificate that the local registry presents, add the `--insecure` option to the command.
      There might be a slight delay before you see a response to the command.

10. Configure the target cluster.

Now that images are mirrored to the local registry, the target cluster must be configured to pull the images from it. Complete the following steps to configure the cluster's global pull secret with the local registry's credentials and then instruct the cluster to pull the images from the local registry.

a. Log in to your Red Hat OpenShift Container Platform cluster:

```
oc login <openshift_url> -u <username> -p <password> -n <namespace>
```

b. Update the global image pull secret for the cluster as explained in the Red Hat OpenShift Container Platform documentation.
Updating the image pull secret provides the cluster with the credentials that are needed for pulling images from your local registry.

Note: If you have an insecure registry, add the registry to the cluster's `insecureRegistries` list by running the following command:

```
oc edit image.config.openshift.io/cluster -o yaml
```

and add the TARGET_REGISTRY to `spec.registrySources.insecureRegistries` as shown in the following example:

```
spec:
  registrySources:
    insecureRegistries:
    - insecure0.svc:5001
    - <TARGET_REGISTRY>
```

If the `insecureRegistries` field does not exist, you can add it.

c. Create the ImageContentSourcePolicy, which instructs the cluster to pull the images from your local registry:

- API Connect:

```
oc apply -f ~/.ibm-pak/data/mirror/$CASE_NAME/$CASE_VERSION/image-content-source-policy.yaml
```

- Foundational services:

```
oc apply -f ~/.ibm-pak/data/mirror/$CS_CASE_NAME/$CS_CASE_VERSION/image-content-source-policy.yaml
```

- EDB:

```
oc apply -f ~/.ibm-pak/data/mirror/$EDB_CASE_NAME/$EDB_CASE_VERSION/image-content-source-policy.yaml
```

d. Verify that each ImageContentSourcePolicy resource was created:

```
oc get imageContentSourcePolicy
```

e. Verify your cluster node status:

```
oc get MachineConfigPool -w
```

Wait for all nodes to be updated before you proceed to the next step.

11. Apply the catalog sources.
Now that the images are mirrored to the target cluster, apply the catalog sources for API Connect and Foundational services. You do not need to apply the catalog source for EDB because the API Connect operator applies that catalog source in its own namespace when the API Connect operand is created.

In the following steps, replace `<Architecture>` with either `amd64`, `s390x` or `ppc64le` according to your environment.

a. Export the variables for the command line to use:

```
export CASE_NAME=ibm-apiconnect
export CASE_VERSION=5.1.0
export ARCH=amd64

export CS_CASE_NAME=ibm-cp-common-services
export CS_CASE_VERSION=5.0.0
export CS_ARCH=amd64
```

b. Generate the catalog source and save it in another directory in case you need to replicate this installation in the future.

i. Get the catalog source:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

cat ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

ii. Get any architecture-specific catalog sources that you need to back up as well:

```
cat ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

You can also navigate to the directory in your file browser to copy these artifacts into files that you can keep for re-use or for pipelines.

c. Apply the catalog sources to the cluster.

i. Apply the universal catalog sources:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources.yaml

oc apply -f ~/.ibm-pak/data/mirror/${CS_CASE_NAME}/${CS_CASE_VERSION}/catalog-sources.yaml
```

ii. Apply any architecture-specific catalog source:

```
oc apply -f ~/.ibm-pak/data/mirror/${CASE_NAME}/${CASE_VERSION}/catalog-sources-linux-${ARCH}.yaml
```

iii. Confirm that the catalog source was created in the `openshift-marketplace` namespace:

```
oc get catalogsource -n openshift-marketplace
```

What to do next

Upgrade your API Connect deployment, following the procedure documented in Upgrading API Connect on OpenShift.

Upgrading API Connect on OpenShift

Upgrade your API Connect installation to a newer version on OpenShift in an online (connected to the internet) environment.

Before you begin

- Review and complete all steps that are documented in Planning your API Connect upgrade on OpenShift and Pre-upgrade preparation and checks on OpenShift.
- If you are upgrading to a version of API Connect that supports a newer version of OpenShift, complete the API Connect upgrade before you upgrade OpenShift.
  Attention: To avoid compatibility issues, do not upgrade your current version of OpenShift until after the API Connect upgrade is complete.

Procedure

1. If you have not run the Pre-upgrade health-checks recently, repeat these checks.
2. Install the cert-manager operator for Red Hat OpenShift by completing the following steps:
   a. Log in to the OpenShift Container Platform web console.

   b. Click Operators > OperatorHub.

   c. In the filter box, type: **`cert-manager Operator for Red Hat OpenShift`**.

   d. Select cert-manager Operator for Red Hat OpenShift and click Install.

   e. On the Install Operator page, complete the following steps:
      i. Update the Update channel if needed. The channel defaults to **`stable-v1`**, which installs the latest stable release of the cert-manager Operator for Red Hat OpenShift.

      ii. Select the Installed Namespace for the operator.
      The default operator namespace is **`cert-manager-operator`**; if that namespace doesn't exist, it is created for you.

      iii. Select an Update approval strategy:
         - Automatic: allow Operator Lifecycle Manager (OLM) to automatically update the operator when a new version is available.
         - Manual: require a user with the appropriate credentials to approve all operator updates.

      iv. Click Install.
   f. Verify the new cert-manager installation by completing the following steps:
      i. Click Operators > Installed Operators.

      ii. Verify that **`cert-manager Operator for Red Hat OpenShift`** is listed with a Status of **`Succeeded`** in the **`cert-manager-operator`** namespace.

      iii. Verify that cert-manager pods are up and running with:

      ```
      oc get pods -n cert-manager
      ```

      For a successful installation, the response looks like the following example:

      ```
      NAME                                       READY   STATUS    RESTARTS   AGE
      cert-manager-bd7fbb9fc-wvbbt               1/1     Running   0          3m39s
      cert-manager-cainjector-56cc5f9868-7g9z7   1/1     Running   0          4m5s
      cert-manager-webhook-d4f79d7f7-9dg9w       1/1     Running   0          4m9s
      ```

   g. Remove the obsolete cert-manager operator (which was provided by IBM Cloud Pak foundational services).
   If you previously deployed the IBM Cloud Pak foundational services operator and the API Connect operator in the same namespace, you might need to manually remove the instance of **`ibm-cert-manager-operator`** that was installed in **`ibm-common-services`**. Complete the following steps to check for the operator and then remove it:

      i. Run the following command to get the list of installed cert-manager operators:

      ```
      oc get pods -A | grep cert-manager-operator
      ```

      If the response lists the **`ibm-cert-manager-operator`** operator, then proceed to the next step to remove it.

      ```
      cert-manager-operator   cert-manager-operator-controller-manager-68ccfc6dd9-2ww4t      2/2     Running     0
      47h
      ibm-common-services     ibm-cert-manager-operator-7c848f5d6c-xtdsb                    1/1     Running     0
      2d
      ```

      ii. Delete the **`ibm-cert-manager-operator`** operator by running the following commands:

      ```
      oc delete subs ibm-cert-manager-operator -n ibm-common-services
      oc get csv  -n ibm-common-services | grep ibm-cert-manager-operator | awk '{ print $1'} | xargs oc delete csv -n
      ibm-common-services
      ```

3. Update the operators for the API Connect deployment.
   API Connect uses three operators; which you update by selecting a newer version for the operator channel. Update the channels in the following sequence:
   a. DataPower: set the channel to v1.9
   b. API Connect: set the channel to v5.1
   c. Foundational services: set the channel to v4.3
      Be sure to update the Foundational services operator channel in all namespaces.

   Complete the following steps to update a channel:

a. Click Operators > Installed Operators.
b. Select the operator to be upgraded.
c. Select the Subscription tab.
d. In the Update channel list, select the new channel version.

If you previously chose automatic subscriptions, the operator version upgrades automatically when you update the operator channel. If you previously chose manual subscriptions, OpenShift OLM notifies you that an upgrade is available. You must manually approve the upgrade before proceeding.

Wait for the operators to update, for the pods to restart, and for the instances to display the **Ready** status. Both the API Connect and DataPower channels must be changed before either operator upgrades. The upgrade of both operators begins when the channel is changed for both operators.

Troubleshooting: If the DataPower operator upgrade fails, see DataPower operator pod stuck waiting for lock removal. If the API Connect operator upgrade fails, see API Connect operator upgrade stuck.

Temporary issue for upgrade from V10.0.6 and V10.0.5: Upgrading to V10.0.7.0 and later involves migrating from Crunchy Postgres to EDB. After the API Connect operator upgrades, the **apiconnectcluster** will be in a pending state, and the **managementcluster** will be in a warning state with the following message:

```
message: management CR is not tracking status, please change CR spec.version
    as soon as possible. Refer https://shortlink.url for details
reason: DatabaseSwitchMode
```

This issue applies only to upgrades from V10.0.6 and V10.0.5. The problem is temporary and resolves when you upgrade the API Connect management subsystem in a later step.

You can check the status from the UI or by running the following command:

```
oc get managementcluster -o yaml
```

4. If you are using the top-level CR (includes Cloud Pak for Integration), then update the top-level **apiconnectcluster** CR:
The **spec** section of the **apiconnectcluster** looks like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-production
  name: prod
  namespace: <APIC_namespace>
spec:
  license:
    accept: true
    use: production
    license: L-MMBZ-295QZQ
  profile: n12xc4.m12
  version: 10.0.7.0
  storageClassName: rook-ceph-block
```

a. Edit the **apiconnectcluster** CR by running the following command:

```
oc -n <APIC_namespace> edit apiconnectcluster
```

b. Update the **spec.version** property to the version you are upgrading to.
c. If you are upgrading to a version of API Connect that requires a new license, update the **spec.license** property accordingly.
For the list of licenses, see API Connect licenses.

d. If you are upgrading from v10.0.6.0 to v10.0.7.0 and were not able to configure an S3 object-store for your management database backups, then remove the **databaseBackup** section, and add the following annotation:

```
metadata:
  annotations:
    apiconnect-operator/backups-not-configured: "true"
```

If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.
e. Optional: If you are upgrading API Connect in Cloud Pak for Integration and want to preserve the Cloud Pak endpoints during the upgrade, add the **deprecatedCloudPakRoute** object to the **spec.management** section of the CR:
Beginning with version 10.0.7.0, API Connect no longer uses the Cloud Pak cpd routes for endpoints when deployed as a component of Cloud Pak for Integration. Instead, the API Management component uses the typical default API Connect routes (or the custom endpoints that are configured in the CR). You can enable the use of Cloud Pak endpoints when you upgrade the API Management component in Cloud Pak for Integration 2023.4.1 or later.

Attention: If you choose to accept the new endpoints (without the Cloud Pak cpd routes), skip this step. However, if you previously configured your own OIDC user registry, be sure to update the **redirect_uri** to reflect the new API Connect endpoint.
To preserve the Cloud Pak endpoints during the upgrade, insert the **deprecatedCloudPakRoute** object into the **spec.management** section as shown in the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  name: <name_of_your_instance>
  namespace: <APIC-namespace>
spec:
  license:
    accept: true
    license: L-MMBZ-295QZQ
    metric: PROCESSOR_VALUE_UNIT
    use: nonproduction
  profile: n1xc17.m48
```

```
  version: 10.0.7.0
  storageClassName: <default-storage-class>
  management:
    deprecatedCloudPakRoute:
      enabled: true
      cloudPakEndpoint:
        hosts:
        - name: cpd-apic.apps.admin-<domain>.com
```

Tip: If you want to maintain the same CPD endpoint, ensure that the `cloudPakEndpoint` hostname is identical to the legacy CPD route name.

f. Optional: If you chose to preserve the Cloud Pak endpoints during the upgrade and want to add a custom certificate for the Cloud Pak route, complete the following steps:

i. Create a Cloud Pak certificate that is signed by a Cloud Pak CA, as in the following example:

```
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
 name: custom-cpd-ca
 namespace: apic
spec:
 selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: custom-cpd
  namespace: apic
spec:
  commonName: small-mgmt-cpd
  dnsNames:
  - cpd-apic.apps.admin-apickeycloak.cp.fyre.ibm.com
  duration: 17520h0m0s
  issuerRef:
    kind: Issuer
    name: custom-cpd-ca
  privateKey:
    algorithm: RSA
    rotationPolicy: Always
    size: 2048
  renewBefore: 720h0m0s
  secretName: custom-cpd
  usages:
  - key encipherment
  - digital signature
  - server auth
```

ii. In the CR, provide the secret name within the `cloudPakEndpoint` property of the new `deprecatedCloudPakRoute` object; for example:

```
spec:
  deprecatedCloudPakRoute:
    enabled: true
    cloudPakEndpoint:
      hosts:
      - name: cpd-apic.apps.admin-<domain>.com
        secretName: custom-cpd
```

g. In the `spec.gateway` section, delete any `template` or `dataPowerOverride` override sections.
If the CR contains an override, then you cannot complete the upgrade.

h. If you have a two data center disaster recovery installation, and are upgrading the warm-standby to V10.0.7.0 or later, then add the annotation:

```
apiconnect-operator/dr-warm-standby-upgrade-data-deletion-confirmation: "true"
```

For more information about two data center disaster recovery upgrade, see Upgrading a two data center deployment on Kubernetes and OpenShift.

i. Save and close the CR to apply your changes.
The response looks like the following example:

```
apiconnectcluster.apiconnect.ibm.com/prod configured
```

Troubleshooting: If you see an error message when you attempt to save the CR, see Troubleshooting upgrade on OpenShift.

j. Run the following command to verify that the upgrade is completed and the status of the top-level CR is `READY`:

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

Important: If you need to restart the deployment, wait until all portal sites complete the upgrade.
After the portal subsystem upgrade is complete, each portal site is upgraded. You can monitor the site upgrade progress from the `MESSAGE` column in the `oc get ptl` output. You can still use the portal while sites are upgrading, although a maintenance page is shown for any sites that are being upgraded. When the site upgrades are complete, the `oc get ptl` output shows how many sites the portal is serving:

```
NAME     READY   STATUS    VERSION          RECONCILED VERSION   MESSAGE              AGE
portal   3/3     Running   <version>        <version>            Serving 2 sites      22h
```

On two data center disaster recovery deployments, the sites are not upgraded until both data centers are upgraded.

5. If you are using individual subsystem CRs: Start with the management subsystem, and update the management CR as follows:

a. Edit the `ManagementCluster` CR:

```
oc edit ManagementCluster -n <mgmt_namespace>
```

b. Update the `spec.version` property to the version you are upgrading to.

c. If you are upgrading to a version of API Connect that requires a new license, update the `spec.license` property accordingly.
For the list of licenses, see API Connect licenses.

d. If you are upgrading from v10.0.6.0 to v10.0.7.0 and were not able to configure an S3 object-store for your management database backups, then remove the `databaseBackup` section, and add the following annotation:

```
metadata:
  annotations:
    apiconnect-operator/backups-not-configured: "true"
```

If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.

e. If you have a two data center disaster recovery installation, and are upgrading the warm-standby to V10.0.7.0 or later, then add the annotation:

```
apiconnect-operator/dr-warm-standby-upgrade-data-deletion-confirmation: "true"
```

For more information about two data center disaster recovery upgrade, see Upgrading a two data center deployment on Kubernetes and OpenShift.

f. Save and close the CR to apply your changes.
The response looks like the following example:

```
managementcluster.management.apiconnect.ibm.com/management edited
```

Troubleshooting: If you see an error message when you attempt to save the CR, see Troubleshooting upgrade on OpenShift.

g. Wait until the management subsystem upgrade is complete before you proceed to the next subsystem. Check the status of the upgrade with: `oc get ManagementCluster -n <mgmt_namespace>`, and wait until all pods are running at the new version. For example:

```
oc get ManagementCluster -n <mgmt_namespace>
NAME          READY   STATUS    VERSION    RECONCILED VERSION   AGE
management    18/18   Running   10.0.7.0   10.0.7.0-1281        97m
```

h. Repeat the process for the remaining subsystem CRs in your preferred order: `GatewayCluster`, `PortalCluster`, `AnalyticsCluster`.
Important: In the `GatewayCluster` CR, delete any `template` or `dataPowerOverride` override sections. If the CR contains an override, then you cannot complete the upgrade.
Important: If you need to restart the deployment, wait until all portal sites complete the upgrade.
After the portal subsystem upgrade is complete, each portal site is upgraded. You can monitor the site upgrade progress from the `MESSAGE` column in the `oc get ptl` output. You can still use the portal while sites are upgrading, although a maintenance page is shown for any sites that are being upgraded. When the site upgrades are complete, the `oc get ptl` output shows how many sites the portal is serving:

```
NAME      READY    STATUS    VERSION          RECONCILED VERSION   MESSAGE              AGE
portal    3/3      Running   <version>        <version>            Serving 2 sites      22h
```

On two data center disaster recovery deployments, the sites are not upgraded until both data centers are upgraded.

6. Upgrade your deployment to the latest version of OpenShift that your current release of API Connect supports.
For more information, see Supported versions of OpenShift.

7. Restart all `nats-server` pods by running the following command:

```
oc -n <namespace> delete pods -l app.kubernetes.io/name=natscluster
```

What to do next

Update your toolkit CLI by downloading it from Fix Central or from the Cloud Manager UI. For more information, see Installing the toolkit.

Troubleshooting upgrade on OpenShift

Review the following troubleshooting tips if you encounter a problem during API Connect upgrade on OpenShift, including as a component of IBM Cloud Pak for Integration (CP4I).

Subsystem is stuck in `Pending` state with a reason of `PreUpgradeCheckInProgress`

Before the subsystem microservices are upgraded, the operator triggers a set of pre-upgrade checks that must pass for the upgrade to proceed. If one or more of the checks fail, the subsystem status remains in `Pending` state with a reason of `PreUpgradeCheckInProgress`. Check the status condition of the subsystem CR to confirm the pre-upgrade check failed. The `status.PreUpgradeCheck` property contains a summary of the failed checks. Full logs for the checks that are carried out can be viewed in the `ConfigMap` referenced in the `status.PreUpgradeCheck` property. The pre-upgrade checks automatically retry until they successfully pass. If you are unable to rectify the problem that causes a check to fail, then open an IBM support case.

Incorrect `productVersion` of gateway pods after upgrade

When you upgrade on OpenShift, the rolling update might fail to start on gateway operand pods due to a gateway peering issue, so that the `productVersion` of the gateway pods is incorrect even though the reconciled version on the gateway CR displays correctly. This problem happens when there are multiple primary pods for gateway peering, and some of the peering pods are assigned to each primary. There can be only one primary pod for gateway-peering; the existence of multiple primaries prevents the upgrade from completing.

Verify that you have a gateway-peering issue with multiple primaries, and then resolve the issue by assigning a single primary for all gateway-peering pods. Complete the following steps:

1. Determine whether multiple primary pods are causing the problem:
   a. Run one of the following commands to check the `productVersion` of each gateway pod.

```
oc get pods -n apic_namespace <gateway_pods> -o yaml | yq .metadata.annotations.productVersion
```

or

```
oc get pods -n apic_namespace <gateway_pods> -o custom-columns="productVersion:.metadata.annotations.productVersion"
```

where:

- *apic_namespace* is the namespace where API Connect is installed
- *<gateway_pods>* is a space-delimited list of the names of your gateway peering pods

b. Verify that the target version of API Connect supports the **productVersion**.
For information about supported versions of DataPower Gateway, see Supported DataPower Gateway versions.

c. If any pod returns an incorrect value for the **productVersion**, check the DataPower operator logs for peering issues; look for messages similar to the following example:

```
"level":"error","ts":"2023-10-20T16:31:47.824751077Z","logger":"controllers.DataPowerRollout","msg":"Unable to check
primaries","Updater.id":"d3858eb3-4a34-4d72-8c51-e4290b6f9cda","Request.Namespace":"apic","Request.Name":"production-
gw-b725a391-4768-4ddc-a73a-d655cf806586","Request.Stage":"mainWork","error":"Multiple primaries found:
map[10.254.14.107:true 10.254.20.175:true]","stacktrace":"github.ibm.com/datapower/datapower-
operator/internal/controllers/datapowerrollout.
(*DataPowerRolloutUpdater).mainWork\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/workers.go:107"}
{"level":"error","ts":"2023-10-20T16:32:17.262171148Z","logger":"controllers.DataPowerRollout","msg":"Attempting to
recover from failed updater","Updater.id":"5524b8b5-8125-492e-aad7-
3e54d107fdbf","Request.Namespace":"apic","Request.Name":"production-gw-b725a391-4768-4ddc-a73a-
d655cf806586","Request.Stage":"checkWorkStatus","error":"Previous updater was unable to complete work: Multiple
primaries found: map[10.254.14.107:true 10.254.20.175:true]","stacktrace":"github.ibm.com/datapower/datapower-
operator/internal/controllers/datapowerrollout.
(*DataPowerRolloutUpdater).checkWorkStatus\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/utils.go:389\ngithub.ibm.com/datapower/datapower-
operator/internal/controllers/datapowerrollout.
(*DataPowerRolloutUpdater).Reconcile\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/reconcile.go:46\ngithub.ibm.com/datapower/datapower-
operator/controllers/datapower.(*DataPowerRolloutReconciler).Reconcile\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/controllers/datapower/datapowerrollout_controller.go:50\nsigs.k8s.io/controller-
runtime/pkg/internal/controller.(*Controller).Reconcile\n\t/home/jenkins/go/pkg/mod/sigs.k8s.io/controller-
runtime@v0.14.5/pkg/internal/controller/controller.go:122\nsigs.k8s.io/controller-runtime/pkg/internal/controller.
(*Controller).reconcileHandler\n\t/home/jenkins/go/pkg/mod/sigs.k8s.io/controller-
runtime@v0.14.5/pkg/internal/controller/controller.go:323\nsigs.k8s.io/controller-runtime/pkg/internal/controller.
(*Controller).processNextWorkItem\n\t/home/jenkins/go/pkg/mod/sigs.k8s.io/controller-
runtime@v0.14.5/pkg/internal/controller/controller.go:274\nsigs.k8s.io/controller-runtime/pkg/internal/controller.
(*Controller).Start.func2.2\n\t/home/jenkins/go/pkg/mod/sigs.k8s.io/controller-
runtime@v0.14.5/pkg/internal/controller/controller.go:235"}
{"level":"error","ts":"2023-10-
20T16:32:17.856155403Z","logger":"controllers_datapowerrollout_gatewaypeering","msg":"Multiple
primaries","DataPowerService":"production-gw","Namespace":"apic","Method":"isPrimary","podName":"production-gw-
2","error":"Multiple primaries found: map[10.254.14.107:true
10.254.20.175:true]","stacktrace":"github.ibm.com/datapower/datapower-
operator/internal/controllers/datapowerrollout/gatewaypeering.
(*GatewayPeeringPrimariesRolloutManager).isPrimary\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/gatewaypeering/gatewaypeering.go:284\ngithub.ibm.com/datapo
wer/datapower-operator/internal/controllers/datapowerrollout/gatewaypeering.
(*GatewayPeeringPrimariesRolloutManager).IsPrimaryByPodName\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/gatewaypeering/gatewaypeering.go:320\ngithub.ibm.com/datapo
wer/datapower-operator/internal/controllers/datapowerrollout.
(*DataPowerRolloutUpdater).mainWork\n\t/home/jenkins/agent/workspace/datapower-
operator_build_v1.6/internal/controllers/datapowerrollout/workers.go:105"}
```

If you confirmed that gateway-peering (multiple primaries) is the issue, complete the following steps to resolve the issue.

2. Attach to a gateway pod:
   a. Run the following command to get a list of gateway pods:

```
oc get pods -l app.kubernetes.io/managed-by=datapower-operator
```

The response looks like the following example:

```
NAME                                                    READY   STATUS    RESTARTS   AGE
datapower-operator-conversion-webhook-585d95cf87-mktrv   1/1     Running   0          2d22h
datapower-operator-f4d8fbd85-5xt47                       1/1     Running   0          2d22h
production-gw-0                                          1/1     Running   0          2d22h
production-gw-1                                          1/1     Running   0          2d22h
production-gw-2                                          1/1     Running   0          2d22h
```

   b. Select a pod from the list and run the following command to attach to it using the NAME:

```
oc -n ns attach -it <pod_NAME> -c datapower
```

3. Select the new primary:
   a. Run the following commands to switch to the API Connect configuration:

```
config; switch apiconnect
```

   b. Run the following command to show the current peering status:

```
show gateway-peering-status
```

Example response:

```
Address      Name              Pending Offset     Link Primary Service port Monitor port Priority
------------ ----------------- ------- ---------- ---- ------- ------------ ------------ --------
10.254.12.45 api-probe         0       19114431   ok   no      16382        26382        2
10.254.12.45 gwd               0       1589848370 ok   no      16380        26380        2
10.254.12.45 rate-limit        0       19148496   ok   no      16383        26383        2
10.254.12.45 ratelimit-module  0       19117538   ok   no      16386        26386        2
10.254.12.45 subs              0       26840742   ok   no      16384        26384        2
```

```
10.254.12.45 tms              0       19115634   ok   no      16381       26381       2
10.254.12.45 tms-external     0       19116159   ok   no      16385       26385       2
10.254.28.61 api-probe        0       19114431   ok   no      16382       26382       1
10.254.28.61 gwd              0       1589849802 ok   yes     16380       26380       1
10.254.28.61 rate-limit       0       19148496   ok   yes     16383       26383       1
10.254.28.61 ratelimit-module 0       19117538   ok   yes     16386       26386       1
10.254.28.61 subs             0       26842695   ok   yes     16384       26384       1
10.254.28.61 tms              0       19117734   ok   yes     16381       26381       1
10.254.28.61 tms-external     0       19117965   ok   yes     16385       26385       1
10.254.36.32 api-probe        0       19114739   ok   yes     16382       26382       3
10.254.36.32 gwd              0       1589848370 ok   no      16380       26380       3
10.254.36.32 rate-limit       0       19148496   ok   no      16383       26383       3
10.254.36.32 ratelimit-module 0       19117538   ok   no      16386       26386       3
10.254.36.32 subs             0       26840742   ok   no      16384       26384       3
10.254.36.32 tms              0       19115634   ok   no      16381       26381       3
10.254.36.32 tms-external     0       19116159   ok   no      16385       26385       3
```

c. Take note of the IP address that has the most primary **yes** options; this IP address will be the new primary for all **functions** involved in gateway peering. In this example, **10.254.28.61** is the primary for the largest number of **functions**; however **api-probe** is not primary on this pod. In the following steps, set **api-probe** as primary on this pod.

4. For every function that needs the primary changed, attach to the primary pod and set the function to primary:
   a. Log out of the current pod by pressing Ctrl+P, and then pressing Ctrl+Q.
   b. Run the following command to attach to a pod that needs its primary updated:

      ```
      oc -n ns attach -it <IP_of_pod> -c datapower
      ```

   c. Run the following command to switch to the API Connect configuration:

      ```
      config; switch apiconnect
      ```

   d. Run the following command to show the current peering status:

      ```
      show gateway-peering-status
      ```

      Verify that the function is set to the wrong primary IP and requires updating (in the example, **api-probe** requires updating).

   e. Run the following command to change the primary for the function:

      ```
      gateway-peering-switch-primary <function>
      ```

      For example:

      ```
      gateway-peering-switch-primary api-probe
      ```

   f. Repeat this process for every function that needs its primary updated.
5. Verify that all functions are now primary on the same IP address by running the following command:

   ```
   show gateway-peering-status
   ```

6. Log out of the gateway pod (Ctrl+P, then Ctrl+Q).

Upgrade fails with an error message about a missing cert-manager

From API Connect V10.0.7.0, it is recommended to use the cert-manager operator for Red Hat OpenShift. If you did not install Red Hat OpenShift cert-manager operator during the upgrade, the upgrade fails with the following error message from the **ibm-apiconnect** pod:

```
cert-manager is required but not installed. Please install a cert-manager operator such as [cert-manager Operator for Red Hat OpenShift]. APIC instance being upgraded
```

Resolve the issue by completing the following steps:

1. Install the cert-manager operator for Red Hat OpenShift:
   a. Log in to the OpenShift Container Platform web console.

   b. Click Operators > OperatorHub.

   c. In the filter box, type: **cert-manager Operator for Red Hat OpenShift**.

   d. Select cert-manager Operator for Red Hat OpenShift and click Install.

   e. On the Install Operator page, complete the following steps:
      i. Update the Update channel if needed. The channel defaults to **stable-v1**, which installs the latest stable release of the cert-manager Operator for Red Hat OpenShift.

      ii. Select the Installed Namespace for the operator.
      The default operator namespace is **cert-manager-operator**; if that namespace doesn't exist, it is created for you.

      iii. Select an Update approval strategy:
      - Automatic: allow Operator Lifecycle Manager (OLM) to automatically update the operator when a new version is available.
      - Manual: require a user with the appropriate credentials to approve all operator updates.

      iv. Click Install.
2. Verify the new cert-manager installation by completing the following steps:
   a. Click Operators > Installed Operators.

   b. Verify that **cert-manager Operator for Red Hat OpenShift** is listed with a Status of **Succeeded** in the **cert-manager-operator** namespace.

   c. Verify that cert-manager pods are up and running with the following command:

```
oc get pods -n cert-manager
```

For a successful installation, the response looks like the following example:

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-bd7fbb9fc-wvbbt               1/1     Running   0          3m39s
cert-manager-cainjector-56cc5f9868-7g9z7   1/1     Running   0          4m5s
cert-manager-webhook-d4f79d7f7-9dg9w       1/1     Running   0          4m9s
```

3. Remove the obsolete IBM cert-manager operator by running the following command:

```
oc delete certmanagers.operator.ibm.com default
```

DataPower operator pod stuck waiting for lock removal

If you see messages in the `datapower-operator` pod log that indicate that the pod is waiting for a lock to be removed:

```
{"level":"info","ts":"2021-03-08T19:29:53.432Z","logger":"leader","msg":"Not the leader. Waiting."}
{"level":"info","ts":"2021-03-08T19:29:57.971Z","logger":"leader","msg":"Leader pod has been deleted, waiting for garbage collection to remove the lock."}
```

The DataPower operator cannot be upgraded until this problem is resolved, see: DataPower operator documentation.

API Connect operator upgrade stuck

If you are upgrading from 10.0.4-ifix3 and the API Connect operator does not begin its upgrade within a few minutes, do the following workaround steps to delete the `ibm-ai-wmltraining` subscription and the associated csv:

1. Run the following command to get the name of the subscription:

```
oc get subscription --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}' -n <APIC_namespace>
```

2. Run the following command to delete the subscription:

```
oc delete subscription <subscription-name> -n <APIC_namespace>
```

3. Run the following command to get the name of the csv:

```
oc get csv --no-headers=true | grep ibm-ai-wmltraining | awk '{print $1}' -n <APIC_namespace>
```

4. Run the following command to delete the csv:

```
oc delete csv <csv-name> -n <APIC_namespace>
```

Deleting the subscription and csv triggers the API Connect operator upgrade.

You see the `denied: insufficient scope` error during an air-gapped deployment

Problem: You encounter the `denied: insufficient scope` message while mirroring images during an air-gapped installation or upgrade.

Reason: This error occurs when a problem is encountered with the entitlement key that is used for obtaining images.

Solution: Obtain a new entitlement key by completing the following steps:

1. Log in to the IBM Container Library.
2. In the Container software library, select Get entitlement key.
3. After the Access your container software heading, click Copy key.
4. Copy the key to a safe location.

Apiconnect operator pod fails

Problem: During installation (or upgrade), the `apiconnect` operator fails with the following message:

```
panic: unable to build API support: unable to get Group and Resources: unable to retrieve the complete list of server APIs:
packages.operators.coreos.com/v1: the server is currently unable to handle the request

goroutine 1 [running]:
github.ibm.com/velox/apiconnect-operator/operator-utils/v2/apiversions.GetAPISupport(0x0)
        operator-utils/v2/apiversions/api-versions.go:89 +0x1e5
main.main()
        ibm-apiconnect/cmd/manager/main.go:188 +0x4ee
```

Additional symptoms:

- Apiconnect operator is in crash loopback status
- Kube `apiserver` pods log the following information:

```
E1122 18:02:07.853093 18 available_controller.go:437] v1.packages.operators.coreos.com failed with:
 failing or missing response from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1:
 bad status from https://10.128.0.3:5443/apis/packages.operators.coreos.com/v1: 401
```

- The IP logged here belongs to the *package server* pod present in the `openshift-operator-lifecycle-manager` namespace
- Package server pods log the following error message: `/apis/packages.operators.coreos.com/v1 API call is being rejected with 401 issue`.

```
E1122 18:10:25.614179 1 authentication.go:53] Unable to authenticate the request due to an error: x509:
certificate signed by unknown authority I1122 18:10:25.614224 1 httplog.go:90]
```

```
verb="GET" URI="/apis/packages.operators.coreos.com/v1" latency=161.243µs resp=401
UserAgent="Go-http-client/2.0" srcIP="10.128.0.1:41370":
```

- The problem is intermittent

Solution:

- If you find the exact symptoms as described, the solution is to delete package server pods in the `openshift-operator-lifecycle-manager` namespace.
- New package server pods log the `200 Success` message for the same API call.

V10.0.6.0: Upgrade stuck in `Pending` state

Upgrades to v10.0.6.0 only.

If your upgrade displays the `Pending` status and seems to be stuck, and the `postgres-operator` pod shows an error like the following example:

```
time="2023-07-17T06:41:09Z" level=error msg="refusing to upgrade due to unsuccessful resource removal"
func="internal/operator/cluster.AddUpgrade()" file="internal/operator/cluster/upgrade.go:134" version=4.7.10
```

You can resolve the problem by deleting the `clustercreate` and `upgrade` pgtasks, which triggers a new attempt at the upgrade.

First, check the logs to verify the error:

1. Run the following command to get the name of the pod:

   ```
   oc get po -n <APIC_namespace> | grep postgres-operator
   ```

2. Run the following command to get the log itself, using the pod name from the previous step:

   ```
   oc logs <pod-name> -c operator -n <APIC_namespace>
   ```

3. If you confirm that the problem is caused by the `resource-removal` issue, delete the `clustercreate` and `upgrade` pgtasks.
   The following example shows the commands to get the pgtask names and then delete the `clustercreate` and `upgrade` pgtasks:

   ```
   oc get pgtasks
   NAME                                          AGE
   backrest-backup-large-mgmt-bce926ab-postgres   36h
   large-mgmt-bce926ab-postgres-createcluster     6d5h
   large-mgmt-bce926ab-postgres-upgrade           36h

   oc delete pgtasks large-mgmt-bce926ab-postgres-createcluster
   pgtask.crunchydata.com "large-mgmt-bce926ab-postgres-createcluster" deleted

   oc delete pgtasks large-mgmt-bce926ab-postgres-upgrade
   pgtask.crunchydata.com "large-mgmt-bce926ab-postgres-upgrade" deleted
   ```

V10.0.6.0: Error message when you save the updated management or APIConnectCluster CR

Upgrades to v10.0.6.0 only.

- Webhook error for incorrect license.
  If you did not update the license ID in the CR, then when you save your changes, the following webhook error might display:

  ```
  admission webhook "vapiconnectcluster.kb.io" denied the request:
  APIConnectCluster.apiconnect.ibm.com "<instance-name>" is invalid: spec.license.license:
  Invalid value: "L-RJON-BYGHM4": License L-RJON-BYGHM4 is invalid for the chosen version version.
  Please refer license document https://ibm.biz/apiclicenses
  ```

  To resolve the error, see API Connect licenses for the list of the available license IDs and select the appropriate license IDs for your deployment. Update the CR with the new license value as in the following example, and then save and apply your changes again.
- Webhook error: `Original PostgreSQL primary not found`. Take the following actions to complete the upgrade and fix the cause of the error message:
  1. If you are using the top-level CR, edit your `apiconnectcluster` CR. If you are using individual subsystem CRs, edit the `ManagementCluster` CR. Add the following annotation:

     ```
     ...
     metadata:
       annotations:
         apiconnect-operator/db-primary-not-found-allow-upgrade=true
         ...
     ```

  2. Continue with the upgrade. When the upgrade is complete, the management CR reports the warning:

     ```
     Original PostgreSQL primary not found, perform db backup and restore to restore original primary.
     ```

  3. Take a new management database backup.
  4. Immediately restore from the new backup taken in the previous step. The action of taking and restoring a management backup results in the establishment of a new Postgres primary, eliminating the CR warning message. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.
- Webhook error: `Original postgres primary is running as replica`. Complete a Postgres failover, see Postgres failover steps. After you apply the Postgres failover steps, the upgrade resumes automatically.

Unexpected behavior in Cloud Manager and API Manager UIs after upgrade

Stale browser cache issues can cause problems after an upgrade. To remedy this problem, clear your browser cache, and open a new browser window.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its

associated management server, but the Git information is based on where it was downloaded from.

Upgrading a two data center deployment on Kubernetes and OpenShift

How to upgrade a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift.

To upgrade API Connect in a 2DCDR deployment, use the upgrade instructions for your platform, but with the extra considerations and steps documented in this topic.

- Upgrading on Kubernetes.
- Upgrading on OpenShift and Cloud Pak for Integration.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Before you begin

- Verify that API Connect is running in both data centers and that replication is working: Verifying replication between data centers.
- Check that you have recent backups of management and portal subsystems: Backup and restore requirements for a 2DCDR deployment.

Key points applicable to both management and portal subsystems

- Your API Connect deployments must be upgraded to the same API Connect release, down to the interim fix level.
- Both data centers must be upgraded in the same maintenance window.
- The warm-standby data center must be upgraded first.
- Upgrade steps might result in an update to the `ingress-ca` X.509 certificate. Extra steps must be taken at various points in the upgrade process to ensure that the `ingress-ca` secret in both data centers contains the same `ingress-ca` X.509 certificate.

Note: If you are not using cert-manager and you customized your certificates, the `ingress-ca` certificate might have a different name. Ensure that the CA certificate that is used by both data centers is the same during all stages of the upgrade.

Upgrading to v10.0.7: Steps for management upgrades

1. Upgrade the warm-standby data center first.
   The upgrade to v10.0.7 involves the temporary deletion of management data on the warm-standby data center. The data is restored to the warm-standby management database when the active management subsystem upgrade is complete.

   Acknowledge that you accept the temporary data deletion from the warm-standby management subsystem, by adding the following annotation to the management CR after the API Connect operator is updated:

   ```
   apiconnect-operator/dr-warm-standby-upgrade-data-deletion-confirmation: "true"
   ```

2. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

   ```
   openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" |
   awk '{print $2}' | base64 -d)
   ```

   if you do not have the `openssl` command available, you can instead run only the `kubectl` part, which produces a larger output:

   ```
   kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
   ```

   if the outputs are different, follow these steps to synchronize the certificates: Synchronizing the ingress-ca X.509 certificate across data centers.
3. Upgrade the active data center.

Upgrading to v10.0.6: Steps for management upgrades

1. Remove the `multiSiteHA` section from the management CRs in both data centers, converting them both to stand-alone deployments.
2. Verify that the management subsystem in both data centers is running as a stand-alone, and that all the pods are running:

   ```
   kubectl get mgmt -n <namespace>

   NAME                          READY   STATUS    VERSION    RECONCILED VERSION   AGE
   <management instance name>    n/n   Running   <version>   <version>-xxxx        14d
   ```

   If one of the data centers returns `6/6` pods then it means that it is still running as a warm-standby, wait a few minutes for it to complete the transition to a stand-alone data center. Run `kubectl describe` and confirm that the `multiSiteHA` section is gone from the `spec` section, and that the `Status` section shows `ha mode` of active in both data centers:

   ```
   kubectl describe mgmt -n <namespace>
   ...
   Status
     Ha Mode:                  active
   ...
   ```

3. Upgrade both data centers.
4. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

   ```
   openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" |
   awk '{print $2}' | base64 -d)
   ```

if you do not have the **openssl** command available, you can instead run only the **kubectl** part, which produces a larger output:

```
kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates: Synchronizing the ingress-ca X.509 certificate across data centers.

5. Add the **multiSiteHA** sections to the management CRs, setting one of them to be the active and the other to be the warm-standby.

Steps for portal upgrades

1. Verify that both data centers have the same **ingress-ca** X.509 certificate in their **ingress-ca** secret. Run the following command in both data centers and check that the output is the same:

   ```
   openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d)
   ```

   if you do not have the **openssl** command available, you can instead run only the **kubectl** part, which produces a larger output:

   ```
   kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
   ```

   if the outputs are different, follow these steps to synchronize the certificates: Synchronizing the ingress-ca X.509 certificate across data centers.
2. Determine which data center has the **ingress-ca** Kubernetes cert-manager certificate object:

   ```
   kubectl get certificates -n <portal namespace> | grep ingress-ca
   ```

   If this data center is the active, failover the portal subsystem to make it the warm-standby data center: Failing over to the warm-standby V10.0.6.0.
3. Start the upgrade of your warm-standby data center by following the upgrade documentation for your platform. Stop at the point where the portal CR is updated with the new API Connect version.
4. Verify that both data centers still have the same **ingress-ca** X.509 certificate, repeating step 1. If they are different, then follow these steps: Synchronizing the ingress-ca X.509 certificate across data centers.
5. Complete the upgrade of your warm-standby data center by updating the portal subsystem CR, following the remaining upgrade steps for your platform.
6. Start the upgrade of your active data center by following the upgrade documentation for your platform. Stop at the point where the portal CR is updated with the new API Connect version.
7. Verify that both data centers still have the same **ingress-ca** X.509 certificate, repeating step 1. If they are different, then follow these steps: Synchronizing the ingress-ca X.509 certificate across data centers.
8. Upgrade the portal subsystem in your active data center by updating the portal subsystem CR, following the remaining upgrade steps for your platform.

How to upgrade when one data center is down

If API Connect is still running on the failed data center, follow the steps that are documented previously to upgrade both data centers before you bring the failed data center back online.

If the failed data center is expected to be down for a long time, you can convert the active data center to a stand-alone data center by following these steps: Removing a two data center deployment, but note the following points:

1. Ensure that the network links to the failed data center are removed.
2. Ensure that the failed data center is set to warm-standby in the **multiSiteHA** section. Do not proceed to the next step until the data center completes the transition to warm-standby. View the status of the management and portal CRs to confirm that **HA Mode** reports "passive".
3. Remove the **multiSiteHA** section from failed data center, and verify that the failed data center resets itself to become an empty stand-alone API Connect deployment (all data is deleted).
4. Before you restore the network links between the data centers, do the following operations:
   - Upgrade API Connect on the failed data center to the same version as the active.
   - Add the **multiSiteHA** sections to both data centers, setting the failed data center to be warm-standby.
     Important: Do not set the failed data center to be active in the **multiSiteHA** section because it results in an overwrite of the data on your working data center with the empty database of your failed data center.

Related concepts

- Two data center warm-standby deployment on Kubernetes and OpenShift

Related tasks

- Installing a two data center deployment on Kubernetes

Upgrading a two data center deployment on Cloud Pak for Integration

How to upgrade a two data center disaster recovery (DR) deployment on Cloud Pak for Integration.

To upgrade a Cloud Pak for Integration deployment of API Connect in a 2DCDR deployment, use the Cloud Pak for Integration upgrade instructions Upgrading on OpenShift and Cloud Pak for Integration, but with the extra considerations and steps documented in this topic.

Before you begin

- Verify that API Connect is running in both data centers and that replication is working: Verifying replication between data centers.
- Check that you have recent backups of management and portal subsystems: Backup and restore requirements for a 2DCDR deployment.

Key points applicable to both management and portal subsystems

- Your API Connect deployments must be upgraded to the same API Connect release, down to the interim fix level.
- Both data centers must be upgraded in the same maintenance window.
- The warm-standby data center must be upgraded first.
- Upgrade steps might result in an update to the `ingress-ca` X.509 certificate. Extra steps must be taken at various points in the upgrade process to ensure that the `ingress-ca` secret in both data centers contains the same `ingress-ca` X.509 certificate.

V10.0.7: 2DCDR upgrade steps for Cloud Pak for Integration

In Cloud Pak for Integration, the `apiconnectcluster` CR (also known as the top-level CR) manages the management and portal subsystems. When the `apiconnectcluster` CR is updated with the new API Connect version, all the subsystems are updated automatically. Follow the upgrade process for Cloud Pak for Integration, review Upgrading on OpenShift and Cloud Pak for Integration, along with the following steps.

1. Verify that both data centers have the same 2DCDR mode for both management and portal subsystems. For example, in DC1, both management and portal subsystems are active, and in DC2, both are warm-standby. If necessary, complete a failover of one of the subsystems so that management and portal have the same mode in the same data centers: Failing over to the warm-standby.
2. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `<apic instance-name>-ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

   ```
   openssl x509 -noout -fingerprint -sha256 -in <(oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d)
   ```

   if you do not have the `openssl` command available, you can instead run only the `oc` part, which produces a larger output:

   ```
   oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
   ```

   if the outputs are different, follow these steps to synchronize the certificates: Synchronizing the ingress-ca X.509 certificate across data centers.
3. Add the warm-standby acceptance of data deletion annotation.
   During the upgrade to V10.0.7.0, data is deleted from the warm-standby management database and is replaced by data from the active data center after its upgrade completes. The upgrade to V10.0.7.0 involves the deletion of management data on the warm-standby data center.

   Because the upgrade process requires that the warm-standby is temporarily without data, you must acknowledge that you accept this data deletion from the warm-standby.

   Add the following annotation to the `apiconnectcluster` CR to acknowledge you accept warm-standby data deletion during upgrade:

   ```
   metadata:
     annotations:
       apiconnect-operator/dr-warm-standby-upgrade-data-deletion-confirmation: "true"
   ```
4. Start the upgrade of your warm-standby data center.
   Follow the upgrade documentation for the API Management capability in https://www.ibm.com/docs/en/cloud-paks/cp-integration. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
5. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
6. Complete the upgrade of your warm-standby data center, updating the operands as described in the Cloud Pak for Integration documentation.
7. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
8. Start the upgrade of your active data center by following the upgrade documentation for the API Management capability in https://www.ibm.com/docs/en/cloud-paks/cp-integration. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
9. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
10. Complete the upgrade of the active data center, updating the operands as described in the Cloud Pak for Integration documentation.
11. Upgrades to V10.0.7 only: 2DCDR installations on Cloud Pak for Integration before V10.0.7 required the creation of a custom endpoint for the management subsystem user-facing certificates. The custom endpoint is not required from V10.0.7, so you can update your management subsystem endpoints.
    In both data centers, edit your top-level CR:

    ```
    oc edit apiconnectcluster
    ```

    Add the following endpoint sections:
    - `spec.management.cloudManagerEndpoint`
    - `spec.management.apiManagerEndpoint`
    - `spec.management.platformAPIEndpoint`
    - `spec.management.consumerAPIEndpoint`

    For example:

    ```
      cloudManagerEndpoint:
        annotations:
          cert-manager.io/issuer: <APIConnectCluster CR name>-ingress-issuer
        hosts:
        - name: <cloudManagerEndpoint>
          secretName: <secret name>
    ```

    where:
    - *<APIConnectCluster CR name>* is the name of your APIConnectCluster CR. Run `oc get apiconnectcluster` to see this name.
    - *<cloudManagerEndpoint>* is the new endpoint that you want to use for the Cloud Manager UI.
    - *<secret name>* name for the secret that stores the TLS certificate for the endpoint. Set this value as follows according to the endpoint:
      - `platformAPIEndpoint`: *<APIConnectCluster CR name>*-mgmt-platform-api
      - `consumerAPIEndpoint`: *<APIConnectCluster CR name>*-mgmt-consumer-api
      - `cloudManagerEndpoint`: *<APIConnectCluster CR name>*-mgmt-admin
      - `apiManagerEndpoint`: *<APIConnectCluster CR name>*-mgmt-api-manager

V10.0.6: 2DCDR upgrade steps for Cloud Pak for Integration

In Cloud Pak for Integration, the `apiconnectcluster` CR (also known as the top-level CR) manages the management and portal subsystems. When the `apiconnectcluster` CR is updated with the new API Connect version, all the subsystems are updated automatically. Follow the upgrade process for Cloud Pak for Integration, starting from Upgrading on OpenShift and Cloud Pak for Integration, along with the following steps.

1. Verify that both data centers have the same `ingress-ca` X.509 certificate in their `<apic instance-name>-ingress-ca` secret. Run the following command in both data centers and check that the output is the same:

   ```
   openssl x509 -noout -fingerprint -sha256 -in <(oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d)
   ```

   if you do not have the `openssl` command available, you can instead run only the `oc` part, which produces a larger output:

   ```
   oc get secret <apic instance-name>-ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
   ```

   if the outputs are different, follow these steps to synchronize the certificates, using the warm-standby as the source data center: Synchronizing the ingress-ca X.509 certificate across data centers.
2. Remove the `multiSiteHA` section from the management sections of the `apiconnectcluster` CRs in both data centers, converting them both to stand-alone management deployments. Be careful to remove the `multiSiteHA` section only from the management sections, and not the portal sections of the `apiconnectcluster` CR.
3. Start the upgrade of your portal warm-standby data center by following the upgrade documentation for the API Management capability in https://www.ibm.com/docs/en/cloud-paks/cp-integration. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
4. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
5. Complete the upgrade of your portal warm-standby data center, updating the operands as described in the Cloud Pak for Integration documentation.
6. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
7. Start the upgrade of your portal active data center by following the upgrade documentation for the API Management capability in https://www.ibm.com/docs/en/cloud-paks/cp-integration. Stop at the point after the API Connect and DataPower operators are updated but before the operands are updated.
8. Repeat step 1 to verify that the `ingress-ca` X.509 certificates are still the same in both data centers.
9. Complete the upgrade of the portal active data center, updating the operands as described in the Cloud Pak for Integration documentation.
10. Add the `multiSiteHA` sections to the management section of the `apiconnectcluster` CRs in both data centers. Set the same active data center for management as for portal.

How to upgrade when one data center is down

If API Connect is still running on the failed data center, follow the steps that are documented previously to upgrade both data centers before you bring the failed data center back online.

If the failed data center is expected to be down for a long time, you can convert the active data center to a stand-alone data center by following these steps: Removing a two data center deployment, but note the following points:

1. Ensure that the network links to the failed data center are removed.
2. Ensure that the failed data center is set to warm-standby in the `multiSiteHA` section. Do not proceed to the next step until the data center completes the transition to warm-standby. View the status of the management and portal CRs to confirm that `HA Mode` reports "passive".
3. Remove the `multiSiteHA` section from failed data center, and verify that the failed data center resets itself to become an empty stand-alone API Connect deployment (all data is deleted).
4. Before you restore the network links between the data centers, do the following operations:
   - Upgrade API Connect on the failed data center to the same version as the active.
   - Add the `multiSiteHA` sections to both data centers, setting the failed data center to be warm-standby.
     Important: Do not set the failed data center to be active in the `multiSiteHA` section because it results in an overwrite of the data on your working data center with the empty database of your failed data center.

Maintaining API Connect

You can use utilities to complete maintenance tasks such as backup, restore, and certificate management on Kubernetes.

Note: When maintaining API Connect, do not use `kubectl exec` or `oc exec` commands to access API Connect pods unless advised by IBM.

- **Advanced configuration**
  You can optionally install and configure advanced features.
- **Backing up, restoring, and disaster recovery**
  Backup your API Connect subsystems, configure scheduled database backups, and recover from a disaster.
- **Backing up, restoring, and disaster recovery V10.0.6.0**
  You can backup and restore API Connect subsystems.
- **Maintaining a two data center deployment**
  How to maintain a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift, including information about normal operation, failure handling, and recovery.
- **Renewing TLS certificates**
  Renew TLS certificates in your API Connect deployment.
- **Monitoring and health checks**
  Monitor your deployment and complete health checks on subsystems.
- **Logging**
  Configure logging for your deployment on Kubernetes.
- **Troubleshooting**
  Troubleshooting problems with your API Connect subsystems.

Advanced configuration

You can optionally install and configure advanced features.

See:

- **Installing License Service**
  License Service helps you assess whether your IBM® API Connect deployment in a Kubernetes environment is compliant with licensing requirements.
- **Installing the Automated API behavior testing application**
  Update the custom resource for installing IBM API Connect to deploy the Automated API behavior testing application on Kubernetes.
- **Installing the toolkit**
- **Changing deployment profiles on Kubernetes and OpenShift**
  You can change an API Connect installation to use different deployment profiles.
- **Changing deployment profiles on OpenShift top-level CR**
  You can change an API Connect top-level CR installation to use different profiles.
- **Replica override for microservices**
  Modify the replica count for microservices by using template overrides for Custom Resources.
- **Changing the deployment profile on Cloud Pak for Integration**
  You can change an API Connect installation to use a different profile.
- **Enabling Developer Portal feature flags on Kubernetes**
  Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- **Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration**
  Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- **Enabling UI feature flags on Kubernetes**
  Use a template override to update the installation CR with settings that control UI features.
- **Enabling UI feature flags on OpenShift and Cloud Pak for Integration**
  Use a template override to update the installation CR with settings that control UI features.
- **Advanced configuration for the management subsystem**
  Specify advanced configuration for the management subsystem.
- **Advanced configuration for the analytics subsystem**
  Specify advanced configuration for the analytics subsystem on Kubernetes, OpenShift, or Cloud Pak for Integration.
- **Advanced configuration for the gateway subsystem**
  Specify advanced configuration for the gateway subsystem.
- **Advanced configuration for the portal subsystem**

Installing License Service

License Service helps you assess whether your IBM® API Connect deployment in a Kubernetes environment is compliant with licensing requirements.

About this task

License Service provides useful features for managing virtualized environments and measuring license utilization. License Service discovers the software that is installed in your infrastructure, helps you to analyze the consumption data, and allows you to generate audit reports. Each report provides you with different information about your infrastructure, for example the computer groups, software installations, and the content of your software catalog.

By default, every License Service audit report presents data from the previous 90 days. You can customize the type and amount of information that is displayed in a report by using filters, and save your personal settings for future use. You can also export the reports to .csv or pdf format, and schedule report emails so that specified recipients are notified when important events occur.

When you install API Connect, the **`license-use`** setting that is configured in the ibm-apiconnect operator CR (custom resource) for a particular subsystem is reflected in the Pod annotations. For the Analytics, Portal, and Management subsystems the **`license-use`** value can be "production" or "nonproduction". For the Gateway subsystem, the **`license-use`** value can be "production", "nonproduction", or "developer". The subsystem pod annotations are scanned by the ibm-licensing-operator, which reports license and CPU usage for each subsystem.

For more information, see License Service in the IBM Cloud Pak foundational services documentation.

Procedure

Install License Service as explained in the installation instructions in License Service in the IBM Cloud Pak foundational services documentation.
The instructions describe the appropriate License Service installation procedure for API Connect deployments, and apply to the Analytics, Portal, Management, and Gateway subsystems.

Installing the Automated API behavior testing application

Update the custom resource for installing IBM® API Connect to deploy the Automated API behavior testing application on Kubernetes.

About this task

You can install the Automated API behavior testing application as part of a new IBM API Connect deployment, or you can add it to an existing deployment:

- Adding the Automated API behavior testing application to an existing deployment
- Including the Automated API behavior testing application in a new deployment

Note: It is not possible to use the Automated API behavior testing application with a two data center disaster recovery configuration (Two data center warm-standby deployment on Kubernetes and OpenShift).

Adding the Automated API behavior testing application to an existing deployment

Before you begin

Make sure that the following requirements are met before attempting to deploy the Automated API behavior testing application:

- The IBM API Connect subsystems are installed.
- The cert-manager is installed for use with IBM API Connectand you are using the default certificates that it generates.
  The Automated API behavior testing application requires that the IBM API Connect deployment use the cert-manager and the default certificates. Custom certificates are not supported and if used, the Automated API behavior testing application will not install correctly.

About this task

Edit the deployed CR (custom resource) for the Management subsystem and add the settings for the Automated API behavior testing application.

Procedure

1. Retrieve the name of the deployed Management subsystem's deployed CR by running the following command:

   ```
   kubectl get managementcluster -n <management_namespace>
   ```

2. Edit the deployed CR by running the following command:

   ```
   kubectl edit managementcluster <management-cr-name> -n <management_namespace>
   ```

3. Append the new `testAndMonitor` definition at the end of the `spec` section, making sure to adhere to the spacing used in the file.
   Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

   ```
   testAndMonitor:
     enabled: true
     hubEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: hub.$STACK_HOST
         secretName: hub-endpoint
     turnstileEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: turnstile.$STACK_HOST
         secretName: turnstile-endpoint
   ```

   `$STACK_HOST` is the ingress subdomain for the API Connect stack, as documented in Installing the Management subsystem.

4. Save the update.
   The Automated API behavior testing application will be installed into the existing Management subsystem.

5. Check the Hub and Turnstile pods periodically until their "READY" values are both "1/1" and their "STATUS" are both "Running".
   Verify that the Hub and Turnstile pods are running, by executing the following command:

   ```
   kubectl get pods -n <management_namespace>
   ```

   Look for pods with names similar to `management-hub-XXXX-XXXX` and `and management-turnstile-XXXX-XXXX`. The following example shows how the response might look:

   ```
   NAME                                    READY   STATUS    RESTARTS   AGE
   management-hub-68745dc4b7-gmlkk         1/1     Running   0          29m
   management-turnstile-b7d978b56-jghvn    1/1     Running   0          24m
   ```

Including the Automated API behavior testing application in a new deployment

Before you begin

Make sure that the cert-manager is installed for use with IBM API Connect and you are using the default certificates that it generates. The Automated API behavior testing application requires that the IBM API Connect deployment use the cert-manager and the default certificates. Custom certificates are not supported and if used, the Automated API behavior testing application will not install correctly.

About this task

Edit the CR (custom resource) for the Management subsystem and add the settings for the Automated API behavior testing application.

Procedure

1. Edit the `ManagementCluster` custom resource (CR) and add the following definition for the Automated API behavior testing application.
   Append the new definition at the end of the `spec` section, making sure to adhere to the spacing used in the file.

   ```
   testAndMonitor:
     enabled: true
     hubEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: hub.$STACK_HOST
         secretName: hub-endpoint
     turnstileEndpoint:
       annotations:
         cert-manager.io/issuer: ingress-issuer
       hosts:
       - name: turnstile.$STACK_HOST
         secretName: turnstile-endpoint
   ```

$STACK_HOST is the ingress subdomain for the API Connect stack, as documented in Installing the Management subsystem.

Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

2. Apply the updated CR by running the following command:

```
kubectl apply -f management_cr.yaml -n <management_namespace>
```

where ***namespace*** is the name of the target installation namespace in the Kubernetes cluster.

The Automated API behavior testing application will be installed with the Management subsystem.

3. Check the Hub and Turnstile pods periodically until their "READY" values are both "1/1" and their "STATUS" are both "Running".
Verify that the Hub and Turnstile pods are running, by executing the following command:

```
kubectl get pods -n <management_namespace>
```

Look for pods with names similar to **management-hub-*XXXX-XXXX*** and **and management-turnstile-*XXXX-XXXX***. The following example shows how the response might look:

```
NAME                                    READY   STATUS    RESTARTS   AGE
management-hub-68745dc4b7-gmlkk         1/1     Running   0          29m
management-turnstile-b7d978b56-jghvn    1/1     Running   0          24m
```

Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just to need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

Procedure

To install and run the toolkit, complete the following steps:

1. Download the toolkit compressed file.
   - To download the toolkit from IBM Fix Central, complete the following steps:
     - Open the IBM Fix Central site in your browser.
     - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
     - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
     - In the Text field, enter `toolkit`, then click Continue.
     - Select the toolkit file that you want to download.
     - Click Continue, then follow the instructions to complete the download operation.
   - To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
     - Open the Cloud Manager or API Manager user interface.
     - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
     - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
     - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the designer_credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 6.
     - To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 5.
     - Close the Install API Connect Toolkit window.
2. Extract the contents of the toolkit compressed file to a folder of your choice.
The contents of the file depend on the your chosen toolkit option and platform, as follows:
   - The apic-slim or apic-slim.exe file is the CLI for IBM API Connect.
   - The apic or apic.exe file is the CLI for IBM API Connect including LoopBack support.
   Tip: If you are using the CLI option, then if you rename the apic-slim file to apic, or the apic-slim.exe file to apic.exe, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
   - The api_designer-*platform* file is the API Designer user interface application for the specified platform.
3. Optional: Delete the **$HOME/.apiconnect/node_clis** directory.
You need to do this only if you replaced a version of **apic** or apic.exe and are using the **apic lb4** command. You need to delete the directory get **apic** to unpack the new loopback.

On Windows, $HOME is defined by environment USERPROFILE.

4. Run the CLI.
   - For the Mac OSX or Linux® platforms, complete the following steps:
     - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Make the CLI file an executable file by entering the following command:

       ```
       chmod +x download_name
       ```

       Where *download_name* is the name of the toolkit file that you downloaded, either apic or apic-slim.
     - Run CLI commands as follows:

       ```
       ./apic command_name_and_parameters
       ```

       or

       ```
       ./apic-slim command_name_and_parameters
       ```

       For details of the CLI commands, see apic.
   - For the Windows platform, complete the following steps:
     - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Run CLI commands as follows:

       ```
       apic command_name_and_parameters
       ```

       or

       ```
       apic-slim command_name_and_parameters
       ```

       For details of the CLI commands, see apic.

   Tip: Add the folder location of your CLI file to your PATH variable so that you can run CLI commands from anywhere in your file system.
5. Install the toolkit credentials.
   If you do not install the toolkit credentials that are provided for download, as detailed previously in the toolkit credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

   ```
   apic client-creds:set toolkit_credentials_file_path/credentials.json
   ```

   where *credentials_file_path* is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.
   Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.
   To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

   ```
   apic client-creds:clear
   ```

   For increased security, an administrator can remove the default credentials from the management server by completing the following steps:
   a. Log in to the management server as an administrator; see Logging in to a management server.
   b. Delete the default credentials by running the following commands:

   ```
   apic registrations:delete toolkit --server mgmt_endpoint_url
   apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
   ```

6. Install the API Designer credentials.
   If you do not install the API Designer credentials that are provided for download, as detailed previously in the API Designer credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the **APIC_DESIGNER_CREDENTIALS** environment variable to the credentials download location, using the mechanism appropriate for your operating system.

   After you have set the **APIC_DESIGNER_CREDENTIALS** environment variable, your API Designer uses the new credentials to authenticate with the management server.

   - Windows:
     If you are using Windows, create an environment variable using one of the following methods:

     - Create a permanent environment variable so that you can start API Designer from any location on your computer:
       - Open the Environment Variables page: Click Start > Settings > System and in the "Related Settings" section of the page, click Advanced System Settings.
       - On the Advanced tab of the System Properties dialog box, click Environment Variables.
       - In the "User variables" section, click New and create an environment variables with the following settings:
         - Variable: **APIC_DESIGNER_CREDENTIALS**
         - Value: *<designer_credentials_file_path>*\designer_credentials.json where ***<designer_credentials_file_path>*** is the location where you stored the designer_credentials.json file.
       - Click OK to save the new environment variable, and then exit the dialog box.
     - If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
       - Open the Windows command prompt.
       - Run the following command to set the temporary environment variable:

         ```
         set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
         ```

         where ***<designer_credentials_file_path>*** is the location where you stored the designer_credentials.json file.

         Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.
       - Now run the following command to start API Designer:

         ```
         C:\"Program Files\API Designer\API Designer.exe"
         ```

By default, API Designer is installed in the C:\"Program Files\API Designer folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

- Mac:
  If you are using Mac OS X, create an environment variable using one of the following methods:
  - Run the following command to set the global environment variable:

    ```
    launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
    ```

    where `<designer_credentials_file_path>` is the location where you stored the API Designer credentials JSON file.
  - Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

    ```
    APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open
    <designer_application_file_path>/'API Designer.app'
    ```

    where:
    - `<designer_credentials_file_path>` is the location to which you downloaded the API Designer credentials JSON file.
    - `<designer_application_file_path>` is the location to which you downloaded and uncompressed the API Designer application.

7. Start the API Designer application from the location to where you stored the extracted file.
   Note:
   - To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
     - In Windows File Explorer, navigate to the USER_HOME\AppData\Local\Programs\api-designer folder.
     - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.
   - To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see Developing your APIs and applications.

For information on using the toolkit CLI, see Using the developer toolkit command-line tool.

Changing deployment profiles on Kubernetes and OpenShift

You can change an API Connect installation to use different deployment profiles.

About this task

After you complete initial installation of the API Connect subsystems, you can change to another of the profiles that are supported for the subsystem. For more information on the available profiles, see: Deployment and component profiles.

Important considerations:

- If you are changing from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (at least 3 worker nodes).
- Based on the resources configured for each microservice in the profile, there might be downtime during the profile change.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Management subsystem:
   a. Verify that subsystem is in `Running` state.

      ```
      kubectl get ManagementCluster
      ```

   b. Edit the CR:

      ```
      kubectl edit ManagementCluster
      ```

      Change `spec.profile` value to the new profile you want to use.

   c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
   d. Verify that subsystem is in `Running` state.

      ```
      kubectl get ManagementCluster
      ```

2. Developer Portal subsystem:
   a. Verify that subsystem is in `Running` state.

      ```
      kubectl get PortalCluster
      ```

   b. Edit the CR:

      ```
      kubectl edit PortalCluster
      ```

      Change `spec.profile` value to the new profile you want to use.

c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in `Running` state.

```
kubectl get PortalCluster
```

3. Analytics subsystem:
Note: If you have a three replica profile, and want to change to one replica, you must first change your analytics configuration to shared storage. Dedicated storage is not supported on the one replica profile.
a. Verify that subsystem is in `Running` state.

```
kubectl get AnalyticsCluster
```

b. Edit the Analytics CR:

```
kubectl edit AnalyticsCluster
```

Change `spec.profile` value to the new profile you want to use.

c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in `Running` state.

```
kubctl get AnalyticsCluster
```

4. Gateway subsystem
a. Verify that subsystem is in `Running` state.

```
kubectl get GatewayCluster
```

b. Edit the CR:

```
kubectl edit GatewayCluster
```

Change `spec.profile` value to the new profile you want to use.

c. When switching from one replica profile to a three replica profile, wait for the microservice to scale up. When switching from a three replica profile to a one replica profile, wait for the microservice to scale down.
d. Verify that subsystem is in `Running` state.

```
kubectl get GatewayCluster
```

Changing deployment profiles on OpenShift top-level CR

You can change an API Connect top-level CR installation to use different profiles.

Before you begin

This task assumes that you have already deployed an API Connect cluster as explained in Deploying on OpenShift and Cloud Pak for Integration. If you are changing from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (3 worker nodes) available. For more information on the available profiles, see: Deployment and component profiles.

Note: If you have a three replica profile, and want to change to one replica, you must first change your analytics configuration to shared storage. Dedicated storage is not supported on the one replica profile.

Procedure

1. Verify that the API Connect cluster is in the `Ready` state by running the following command:

```
oc get apiconnectcluster
```

2. Update the API Connect Cluster CR:
a. Open the API Connect Cluster CR in edit mode by running the following command:

```
oc edit apiconnectcluster <APIC-instance-name>
```

b. In the CR, change the `spec.profile` value to the profile that you want to use.
c. Save the change by running the following command: `:wq`
3. Wait for microservice to scale up, or scale down, as needed.
4. Verify that the API Connect cluster is in the `Ready` state by running the following command:

```
oc get apiconnectcluster
```

Replica override for microservices

Modify the replica count for microservices by using template overrides for Custom Resources.

When API Connect is deployed, the deployment profile you select drives the number of replicas to be used for most deployment. For one replica profiles, specified by `n1x....`, the replica count is 1. For three replica profiles, specified by `n3x...`, replica count is 3. Note, however, that for Cloud Pak for Integration the API Connect Cluster profile `n3xc4.m16` replica count is 1.

You can override the replica count for individual deployments by using template overrides.

- Template overrides are a piece of yaml that should be added to the Kubernetes custom resource. The examples provided in this topic show only that the `template:` element should be under the main `spec:` element of a CR. The rest of the CR is not shown.
- Specification of microservice overrides in CP4I requires prefixing the microservice name with a designator for the subsystem type, as follows:
  - Management subsystem: `mgmt-`
  - Gateway: `gw-`
  - Analytics: `a7s-`

  For example, for the `apim` Management microservice, use `apim` when defining the override for OpenShift (or Kubernetes), but use `mgmt-apim` for a CP4I override.

Management subsystem

For a Management CR in Kubernetes or OpenShift, the following override specifies 5 replicas for the `apim` deployment:

```
spec:
  template:
  - name: apim
    replicaCount: 5
```

You can also configure replicate count through a similar template override for:

- `juhu`
- `ldap`
- `lur`

The same override for CP4I:

```
spec:
  template:
  - name: mgmt-apim
    replicaCount: 5
```

On CP4I, you can also configure replicate count through a similar template override for:

- `mgmt-juhu`
- `mgmt-ldap`
- `mgmt-lur`

Gateway subsystem

The following override is supported for a Gateway CR in Kuberneter or OpenShift:

```
spec:
 template:
  - name: datapower
    replicaCount: 5
```

The same override for CP4I:

```
spec:
  template:
  - name: gw-datapower
    replicaCount: 5
```

Analytics subsystem

The Analytics microservices that can be scaled using template override are:

- `director`
- `ingestion`
- `mtls-gw`
- `storage`
  Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

On CP4I, you can also configure replica count through a similar template override for:

- `a7s-directory`
- `a7s-ingestion`
- `a7s-mtls-gw`
- `a7s-storage`
  Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

Changing the deployment profile on Cloud Pak for Integration

You can change an API Connect installation to use a different profile.

Before you begin

This task assumes that you have already deployed an API Connect cluster as explained in Deploying on OpenShift and Cloud Pak for Integration. If you are switching from a one replica profile to a three replica profile, ensure that you have three replica profile hardware (3 worker nodes) available. For more information on the available profiles, see: Deployment and component profiles.

Note: If you have a three replica profile, and want to change to one replica, you must first change your analytics configuration to shared storage. Dedicated storage is not supported on the one replica profile.

Procedure

1. Verify that the API Connect cluster is in the **Ready** state.
   Open the IBM Cloud Pak Platform UI, locate the API Management capability on the Integration instances tab, and verify that the state displays as **Ready**. Leave the page open for the next step.
2. Update the API Management instance:
   a. In the Platform UI, click the overflow menu next to the instance and select Edit.
   b. In the Details section of the deployment pages, navigate to the Deployment profile field.
   c. Click the field and select the new deployment profile.
3. Click Update to save the change.
4. Wait for microservice to scale up, or scale down, as needed.
5. Verify that the status for the API Management instance displays as **Ready**.

Enabling Developer Portal feature flags on Kubernetes

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description
PORTAL_SKIP_WEB_ENDPOINT_VALIDATION	**true**	**www**	**admin**	Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible. Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the **portal-www** pod, and **admin** container logs: **An error occurred contacting the provided portal web endpoint: example.com** **The provided Portal web endpoint example.com returned HTTP status code 504** You can disable the check by setting this flag to **false**.
ENABLE_TRUSTED_REVERSE_PROXY	**false**	**www**	**admin**	Controls whether the Drupal **trusted_reverse_proxy** module can manage the reverse proxies list. If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal **trusted_reverse_proxy** module, and setting this feature flag to **true**. The Drupal module inspects the **x-forwarded-for** headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in **X-Forwarded-For: <client>, <proxy1>, <proxy2>**, the client IP **<client>** will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy. To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the **x-forwarded-for** header is passed through to the **portal-www** pod. For example, the default behavior of the **nginx-ingress-controller** is to ignore the inbound **x-forwarded-for** header, and create a new one. To change this behavior, update the **nginx-ingress-controller** **configmap** with the following information: **data:** **compute-full-forwarded-for: "true"** **use-forwarded-headers: "true"** Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the **x-forwarded-for** header.

Procedure

1. If API Connect is already deployed, retrieve the name of the deployed CR for the Developer Portal subsystem by running the following command:

   kubectl get portalcluster -n *<portal_namespace>*

   If API Connect yet is not yet deployed, skip this step.

2. Edit the **portal** CR by running the following command:

```
kubectl edit portalcluster <portal-cr-name> -n <portal_namespace>
```

3. In the editor, append the feature flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.
   The following example enables the `trusted_reverse_proxy` feature.

```
spec:
...
  template:
  - containers:
    - env:
        - name: ENABLE_TRUSTED_REVERSE_PROXY
          value: true
        name: container
      name: pod
```

   Where:
   - `container` is the name of the portal container.
   - `pod` is the name of the portal pod.

   Note: You can add multiple feature flags to the CR.
4. Save the update.
   The Developer Portal subsystem is updated to enable or disable features as specified.

Enabling Developer Portal feature flags on OpenShift and Cloud Pak for Integration

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description

`PORTAL_SKIP_WEB_ENDPOINT_VALIDATION`	`true`	`www`	`admin`	Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible.
				
				Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the `portal-www` pod, and `admin` container logs:
				
				`An error occurred contacting the provided portal web endpoint: example.com`
				`The provided Portal web endpoint example.com returned HTTP status code 504`
				
				You can disable the check by setting this flag to `false`.

`ENABLE_TRUSTED_REVERSE_PROXY`	`false`	`www`	`admin`	Controls whether the Drupal `trusted_reverse_proxy` module can manage the reverse proxies list.
				
				If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal `trusted_reverse_proxy` module, and setting this feature flag to `true`.
				
				The Drupal module inspects the `x-forwarded-for` headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in `X-Forwarded-For: <client>, <proxy1>, <proxy2>`, the client IP `<client>` will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy.
				
				To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the `x-forwarded-for` header is passed through to the `portal-www` pod. For example, the default behavior of the `nginx-ingress-controller` is to ignore the inbound `x-forwarded-for` header, and create a new one. To change this behavior, update the `nginx-ingress-controller` `configmap` with the following information:
				
				`data:`
				`  compute-full-forwarded-for: "true"`
				`  use-forwarded-headers: "true"`
				
				Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the `x-forwarded-for` header.

Procedure

1. If API Connect is already deployed, retrieve the name of the deployed CR by running the following command:

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

   If API Connect yet is not yet deployed, skip this step.

2. Edit the `apiconnectcluster` CR by running the following command:

   ```
   oc edit apiconnectcluster <APIC_instance_name> -n <APIC_namespace>
   ```

3. In the CR, append the feature flag definition to the end of the `spec.portal` section, making sure to adhere to the spacing used in the file.
   The following example enables the `trusted_reverse_proxy` feature.

   ```
   spec:
   ...
     portal:
       template:
       - containers:
         - env:
            - name: ENABLE_TRUSTED_REVERSE_PROXY
              value: true
           name: container
         name: pod
   ```

   Where:
   - *container* is the name of the portal container.
   - *pod* is the name of the portal pod.

   Note: You can add multiple feature flags to the CR.
4. Close and save the file with the following command:

   ```
   :wq
   ```

   The Developer Portal subsystem is updated to enable or disable features as specified.

Enabling UI feature flags on Kubernetes

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
`ffAdvancedPublish`	`true`		Enable advanced publish option (deprecated)
`allOmniSearch`	`true`		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
`draftsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for drafts list
`catalogProductsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for catalog
`orgsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for org lists
`appsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for apps
`subscriptionsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
`requestedApprovalsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for requested approvals
`approvalTasksOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for approvals
`productApisOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for apis within products
`membersOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for member lists

Procedure

1. If you already deployed API Connect, retrieve the name of the deployed CR for the Management subsystem by running the following command:

   ```
   kubectl get managementcluster -n <management_namespace>
   ```

   If you did not deploy API Connect yet, skip this step.

2. Edit the `management` CR by running the following command:

   ```
   kubectl edit managementcluster <management-cr-name> -n <management_namespace>
   ```

3. In the editor, append the feature-flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.
   The following example enables the OmniSearch feature on all possible UI pages.

   ```
   spec:
     template:
       - name: ui
         containers:
         - name: ui
           env:
             - name: APIC_UI_FEATURES
               value: allOmniSearch=always
   ```

   `APIC_UI_FEATURES` is a URL-encoded string. Each parameter is formed as `featureId=$featureValue`, and you can list multiple parameters with the `&` delimiter (`featureId=$featureValue&featureId=$featureValue`).

Use the **featureId** values from Table 1. For each feature, specify one of the following **featureValue** settings:

- **true** - On by default, but can be overridden by the user, for their browser alone
- **false** - Off by default, but can be overridden by the user, for their browser alone
- **never** - Off regardless of browser setting, disabled in feature flag page
- **always** - On regardless of browser setting, disabled in feature flag page

4. Save the update.
   The cluster is updated to enable or disable features as specified.

Enabling UI feature flags on OpenShift and Cloud Pak for Integration

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
ffAdvancedPublish	true		Enable advanced publish option (deprecated)
allOmniSearch	true		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
draftsOmniSearch	true	allOmniSearch	Enable OmniSearch for drafts list
catalogProductsOmniSearch	true	allOmniSearch	Enable OmniSearch for catalog
orgsOmniSearch	true	allOmniSearch	Enable OmniSearch for org lists
appsOmniSearch	true	allOmniSearch	Enable OmniSearch for apps
subscriptionsOmniSearch	true	allOmniSearch	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
requestedApprovalsOmniSearch	true	allOmniSearch	Enable OmniSearch for requested approvals
approvalTasksOmniSearch	true	allOmniSearch	Enable OmniSearch for approvals
productApisOmniSearch	true	allOmniSearch	Enable OmniSearch for apis within products
membersOmniSearch	true	allOmniSearch	Enable OmniSearch for member lists

Procedure

1. If you already deployed API Connect, retrieve the name of the deployed CR by running the following command:

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

   If you did not deploy API Connect yet, skip this step.

2. Edit the **apiconnectcluster** CR by running the following command:

   ```
   oc edit apiconnectcluster <APIC_instance_name> -n <APIC_namespace>
   ```

3. In the CR, append the feature-flag definition to the end of the **spec.management** section, making sure to adhere to the spacing used in the file.
   The following example enables the OmniSearch feature on all possible UI pages.

   ```
   spec:
     management:
       template:
         - name: ui
           containers:
           - name: ui
             env:
               - name: APIC_UI_FEATURES
                 value: allOmniSearch=always
   ```

   APIC_UI_FEATURES is a URL-encoded string. Each parameter is formed as **featureId=$featureValue**, and you can list multiple parameters with the **&** delimiter (**featureId=$featureValue&featureId=$featureValue**).

   Use the **featureId** values from Table 1. For each feature, specify one of the following **featureValue** settings:

   - **true** - On by default, but can be overridden by the user, for their browser alone
   - **false** - Off by default, but can be overridden by the user, for their browser alone
   - **never** - Off regardless of browser setting, disabled in feature flag page
   - **always** - On regardless of browser setting, disabled in feature flag page

4. Close and save the file with the following command:

   ```
   :wq
   ```

   The cluster is updated to enable or disable features as specified.

Advanced configuration for the management subsystem

Specify advanced configuration for the management subsystem.

See:

- **Updating management subsystem password, keys, and secrets**
  Update the cloud administrator password, the key for encrypted database fields, and the `client_ID` and `client_secret` values for the management subsystem in an API Connect deployment.
- **Configuring maximum size of client requests to the Management subsystem**
  You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.
- **Setting rate limits for public APIs on the management service for a Kubernetes environment**
  Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.
- **Enabling API governance on Kubernetes**
  You can optionally configure API governance in API Connect on a Kubernetes, OpenShift, or IBM® Cloud Pak for Integration deployment by enabling the governance microservice.
- **Configuring monetization on Kubernetes**
  You can optionally configure monetization in API Connect on a Kubernetes deployment by enabling the billing microservice.
- **Overriding resources for Postgres components on Kubernetes V10.0.6.0**
  You can optionally override the CPU and Memory resources used by postgres database components.
- **Overriding resources for Postgres components on OpenShift V10.0.6.0**
  You can optionally override the CPU and Memory resources used by postgres database components.
- **Restarting management subsystem pods**
  How to restart management subsystem pods in your API Connect deployment.

Updating management subsystem password, keys, and secrets

Update the cloud administrator password, the key for encrypted database fields, and the `client_ID` and `client_secret` values for the management subsystem in an API Connect deployment.

- **Changing the cloud administrator password**
  Update the cloud administrator password for the management subsystem using the Cloud Manager UI, the toolkit CLI, or an API call.
- **Changing the database encryption key**
  Replace the value of the secret that is used for encrypting database fields in the management subsystem.
- **Changing the registration client_id and client_secret for applications**
  Change the registration credentials (the `client_id` and `client_secret`) for applications provided by the management subsystem.

Changing the cloud administrator password

Update the cloud administrator password for the management subsystem using the Cloud Manager UI, the toolkit CLI, or an API call.

About this task

In the management subsystem for an API Connect on-premises deployment, the administrator is called the *cloud administrator* and works with the management server to configure the deployment of the local server cloud. Updating the cloud administrator password for the management subsystem requires no service interruption because the change takes place immediately.

You can change the cloud administrator's password using any of the following methods.

Note: Passwords must contain eight or more characters, with at least one character from three of the following categories: lowercase letters, uppercase letters, numerals, and punctuation (for example: ! $ # %). The same character cannot be used more than twice consecutively.

Cloud Manager UI

1. Log in to the Cloud Manager UI.
2. Change your Cloud Manager password.

Toolkit CLI

1. Run the `apic login` command to log in to a management server.
2. Run the `apic me:change-password` command to change your password.

API call

1. Run the `apic login` command to log in to a management server.
   You must log in to a management server with the toolkit CLI before you can call the API Connect APIs.
2. Execute the `/api/me/change-password` API call. as explained in the "Resource: Me" section of the IBM API Connect Platform - Cloud Management API.

The next time that you log in as the cloud administrator, you must use the new password.

Changing the database encryption key

Replace the value of the secret that is used for encrypting database fields in the management subsystem.

Before you begin

Take a backup of your existing management database encryption key. If you need to restore a management database backup, you must have the encryption key that was in use at the time the database backup was created.

To take backup of your encryption key, run the following command:

```
kubectl -n <management namespace> get secret <management enc key> -o yaml > management_enc_key_backup_<date>.yaml
```

The key is backed up to the file management_enc_key_backup_*<date>*.yaml. Set *<date>* to the current date, for example management_enc_key_backup_20240101.yaml, so that you can identify when this backup was taken.
Keep your backed up encryption secret with your management infrastructure configuration backup: Taking a management infrastructure configuration backup. If you want to restore a database backup that was taken when a previous encryption key was in use, then see restore backup.

About this task

This update does not require an outage; however some operations might take longer than usual due to database locking.
Note: If you installed API Connect on VMware, do not attempt to follow these steps. If you want to update the management encryption secret on VMware, then open a support case.

Procedure

1. Create a file called rotate_mgmt_enc_secret.yaml, and paste in the following contents:

   ```
   apiVersion: management.apiconnect.ibm.com/v1beta1
   kind: ManagementSecretRotation
   metadata:
     name: rotate-secret
   spec:
     managementCluster: <name of management CR>
     rotateEncryptionSecret:
       rotate: true
   ```

   where *<name of management CR>* can be found by running:

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

   Note: If you upgraded from V2018, the database encryption key was carried forward and is called ***managementUpgradeName*-encryption-secret**.
2. Run the following command to apply the **rotate-secret** CR:

   ```
   kubectl -n <namespace> create -f rotate_mgmt_enc_secret.yaml
   ```

   The update can take a few minutes. You can monitor the progress with:

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

   Shortly after you run the **kubectl create** command, you should see the **rotate-secret** job:

   ```
   NAME                                                    COMPLETIONS   DURATION   AGE
   ...
   apic-mgmt-ec78457-rotate-secret                         1/1           22s        9m21s
   ...
   ```

What to do next

If you need to restore a management database backup that was taken before you updated the encryption key, then you must restore your previous encryption key from your backup.

1. Edit your secret backup file: management_enc_key_backup_*<date>*.yaml and change **metadata.name** to a unique name for your backup secret. Delete all other **metadata** contents. Your updated YAML file should look like this:

   ```
   apiVersion: v1
   data:
     encryption_secret.bin:
   If7HlVzZeAgrYrwvcEhv1Ub0LVXNQsN0mGISy1GvYXKA4d1tSWLB80hB89sn+VvN0/vAVTU90AP6jPOsi5LpOaddfoqT85ugvjEwHLnEtSI6CbEr05NtMUwJGA
   r5/KWfjj6mwCn+EMp4VBsEDppU54yvycyb7iBLRtGyrPktTaw=
   kind: Secret
   metadata:
     name: backup_mgmt_encryption_secret
   type: Opaque
   ```

2. Restore the encryption secret from your updated backup file:

   ```
   kubectl -n <management namespace> create -f management_enc_key_backup_<date>.yaml
   ```

3. Verify that the backup secret was created:

   ```
   kubectl -n <management namespace> get secrets
   ```

   You should see your restored backup secret in the output:

   ```
   NAME                                TYPE             DATA   AGE
   ...
   backup_mgmt_encryption_secret       Opaque           3      10s
   ...
   ```

4. In your mgmtrestore_cr.yaml file, add the property **spec.mgmtEncKeyName** and set it to your newly created backup secret name:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management- # this the prefix for the name of the restore CR that is created
spec:
  backupName: <backup CR name>
  subsystemName: <management CR name>
  mgmtEncKeyName: backup_mgmt_encryption_secret
```

The encryption key that is contained in **backup_mgmt_encryption_secret** is copied to your active encryption secret. After you restore the backup, you might want to update the encryption key again (for the same reason that you previously updated the secret).

For more information about management database backup restore, see Restoring from a management database backup.

Changing the registration client_id and client_secret for applications

Change the registration credentials (the **client_id** and **client_secret**) for applications provided by the management subsystem.

About this task

Update the application registration **client_id** and **client_secret** by changing the Kubernetes secret containing them. The **apim** file-watcher detects the change and automatically updates the following secrets in the database:

- toolkit: **management-cli-cred** (Customer facing)
  When you update the secret for the toolkit, users must download and install new toolkit credentials as explained in Installing the toolkit.
- consumer-toolkit: **management-ccli-cred** (Customer facing)
  When you update the secret for the toolkit, users must download and install new toolkit credentials as explained in Installing the toolkit.
- designer: **management-dsgr-cred** (Customer facing)
  When you update the secret for API Designer, users must download and install new credentials as explained in Installing the toolkit.
- atm: **management-atm-cred** (Internal)
- governance: **management-governance-cred** (Internal)
- consumer-ui: **management-cui-cred** (Internal)
- juhu: **management-juhu-cred** (Internal)
- ui: **management-ui-cred** (Internal)
  The API Manager and Cloud Manager read the credentials from the back-end on start-up.

When you update registration credentials, the registration name that is associated with the updated secret changes. Table 1 lists the registration secrets with the default registration name and the pattern used for generating updated registration names for each secret. The new registration names follow the patterns shown in the table. The second and subsequent updates append a number to the name; for example, **toolkit-application-registration**, **toolkit-application-registration0** and then **toolkit-application-registration1**.

Table 1. Secret names and the corresponding default, and new, registration names for each secret

Secret	Default registration name	New registration name pattern
management-cli-cred	toolkit-default	toolkit-application-registration*
management-ccli-cred	consumer-toolkit-default	consumer-toolkit-application-registration*
management-dsgr-cred	designer-default	designer-registration*
management-atm-cred	atm-default	atm-default*
management-governance-cred	compliance-default	compliance-default*
management-cui-cred	consumer-ui-default	consumer-ui-default*
management-juhu-cred	juhu-default	juhu-default*
management-ui-cred	ui-default	ui-default*

Procedure

1. To renew an internal certificate, complete the following steps:
   a. Create a ManagementSecretRotation CR (for example, operationcr.yaml) with contents similar to the following example, and list the secrets that you want to update:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementSecretRotation
metadata:
  name: rotate-credentials
spec:
  managementCluster: management
  rotateClientIDandSecret:
    secrets:
    - management-ui-cred
    - management-cui-cred
    - management-juhu-cred
    - management-atm-cred
```

   b. Apply the CR by running the following command:

```
kubectl create -f operationcr.yaml -n namespace
```

   where:

   - ***namespace*** is the namespace where you installed the management subsystem
   - operationcr.yaml is the file the contains the CR

Applying the resource updates the credential for the specified secrets with new values.

2. To update customer-facing credentials, complete the following steps:
   a. For each of the credential secrets, run the following command to update the **client_id** and **client_secret**:

      ```
      kubectledit secret credential_secret_name -n namespace
      ```

      where:
      - ***namespace*** is the namespace where the management subsystem is installed.
      - ***credential_secret_name*** is the name of the secret that you want to update.

      The response opens the secret in a text editor as shown in the following example:

      ```
      apiVersion: v1
      data:
        credential.json:
      eyAiaWQiOiAiNTY3MTYyNjYtZDE0Zi00ODZmLWI0MWYtZTE3MzZkZTg3NjE0IiwgInNlY3JldCI6ICJmYjU5MDdhNy0zMjZjLTQ4ZWYtYjQ0ZC1hMTI5Z
      mZkMzNiMTEiIH0=
      ```

      In the secret, the value of **credential.json** is a base64 encoded string of the following JSON format. Decode it first, modify the values of **id** and **secret**, and encode back as base64 and paste it in the secret. (Note that either **id** or **secret** or both can be updated, but if the credentials have been compromised then its recommended to update both id and secret). For example:

   b. Decode the value of **credential.json**.
      To decode the value, copy it from the secret in step 2a, and paste it between the quotation marks as the ***encoded_value*** in the following command:

      ```
      echo "encoded_value" | base64 --decode
      ```

      Using the value returned in step 1, the command looks like the following example:

      ```
      echo
      "eyAiaWQiOiAiNTY3MTYyNjYtZDE0Zi00ODZmLWI0MWYtZTE3MzZkZTg3NjE0IiwgInNlY3JldCI6ICJmYjU5MDdhNy0zMjZjLTQ4ZWYtYjQ0ZC1hMTI5
      ZmZkMzNiMTEiIH0=" | base64 --decode
      ```

      The response displays decoded values of the **id** and **secret**, as in the following example:

      ```
      { "id": "56716266-d14f-486f-b41f-e1736de87614", "secret": "fb5907a7-326c-48ef-b44d-a129ffd33b11" }
      ```

   c. Update the values of **id** and **secret**.
      You can use any string that meets the following requirements:
      - The **id** (it becomes **client_id**) must be unique across all credentials in your entire deployment.
      - The values must be enclosed in straight quotation marks. To make sure you use the correct character, create the values in a plain text editor.

   d. Encode the updated **id** and **secret** as base64 by running the following command:

      ```
      echo -n '{ "id": "new_id", "secret": "new_secret" }' | base64
      ```

      Replace ***new_id*** and ***new_secret*** with your updated values. For example:

      ```
      echo -n '{ "id": "21034840-b439-a393-4482-a948bcb39603", "secret": "43450ab9-4038-acf0-0498-793b29ade396" }' | base64
      ```

      Returns the following base640-encoded value:

      ```
      aWQ6IDIxMDM0ODQwLWI0MzktYTM5My00NDgyLWE5NDhiY2IzOTYwMywgc2VjcmV0OiA0MzQ1MGFiOS00MDM4LWFjZjAtMDQ5OC03OTNiMjlhZGUzOTY=
      ```

   e. Paste the newly coded value for **credential.json** into the secret.
   f. Save the secret.
      When you save the secret, the logs in the **apim** pods indicate that a new registration was created. For example, if the **client_id** changed, the display looks similar to the following messages

      ```
      2020-09-14T09:55:58.331Z bhendi:audit [ce64fd50-e4de-4cfb-baa7-7379a892b2c9] Created internal resource "Registration"
      'toolkit-application-registration-0 (Toolkit Application Registration)' (id=ffc3c398-a008-403a-aa0a-218604dc99c2,
      url=/api/cloud/registrations/ffc3c398-a008-403a-aa0a-218604dc99c2, namespace=cloud)
      2020-09-14T09:55:58.331Z audit [ce64fd50-e4de-4cfb-baa7-7379a892b2c9] Creating a new registration for type, toolkit,
      with client_id: toolkit-client-id-2.
      ...
      2020-09-14T09:58:15.375Z bhendi:audit [b3edcdd8-2d55-445f-a386-5b92068bf5bd] Created internal resource "Registration"
      'consumer-toolkit-application-registration (Consumer Toolkit Application Registration)' (id=3b51aba6-c365-4fc0-bc4f-
      371de99d2c18, url=/api/cloud/registrations/3b51aba6-c365-4fc0-bc4f-371de99d2c18, namespace=cloud)
      2020-09-14T09:58:15.375Z audit [b3edcdd8-2d55-445f-a386-5b92068bf5bd] Creating a new registration for type,
      consumer_toolkit, with client_id: consumer-toolkit-client-id.
      ...
      2020-09-14T10:42:40.464Z bhendi:audit [f1036cb9-a255-445f-a655-35cfb8de317c] Created internal resource "Registration"
      'designer-registration (Designer Registration)' (id=c44b51a7-731d-42d4-ad93-961ebbd6f5b0,
      url=/api/cloud/registrations/c44b51a7-731d-42d4-ad93-961ebbd6f5b0, namespace=cloud)
      2020-09-14T10:42:40.464Z audit [f1036cb9-a255-445f-a655-35cfb8de317c] Creating a new registration for type, designer,
      with client_id: designer-client-id.
      ...
      2020-09-14T10:50:38.377Z bhendi:audit [28336f83-622d-4928-8b2a-3597991e3842] Created internal resource "Registration"
      'test-and-monitor (Test and Monitor)' (id=a9810cc6-bb54-4ab5-be62-3833038df26e,
      url=/api/cloud/registrations/a9810cc6-bb54-4ab5-be62-3833038df26e, namespace=cloud)
      2020-09-14T10:50:38.377Z audit [28336f83-622d-4928-8b2a-3597991e3842] Creating a new registration for type, atm, with
      client_id: atm-client-id.
      ...
      2020-09-14T10:57:11.377Z bhendi:audit [490181e1-017f-49cf-83dc-58fe148ec1cb] Created internal resource "Registration"
      'consumer-user-interface-registration (Consumer User Interface Registration)' (id=5be9e542-1424-4a5b-877e-
      dd28da898f58, url=/api/cloud/registrations/5be9e542-1424-4a5b-877e-dd28da898f58, namespace=cloud)
      2020-09-14T10:57:11.377Z audit [490181e1-017f-49cf-83dc-58fe148ec1cb] Creating a new registration for type,
      consumer_ui, with client_id: cui-client-id.
      ...
      2020-09-14T10:58:53.404Z bhendi:audit [298b3940-23b1-4ed5-b8a7-532af66699c5] Created internal resource "Registration"
      'juhu-application-registration (Juhu Application Registration)' (id=92698052-aca0-4d06-8ad3-21ed26a13734,
      url=/api/cloud/registrations/92698052-aca0-4d06-8ad3-21ed26a13734, namespace=cloud)
      2020-09-14T10:58:53.404Z audit [298b3940-23b1-4ed5-b8a7-532af66699c5] Creating a new registration for type, juhu,
      with client_id: juhu-client-id.
      ```

```
...
2020-09-14T11:00:52.377Z bhendi:audit [866bb70b-3bb5-4ac3-9a59-0b60ade7feb2] Created internal resource "Registration"
'user-interface-registration (User Interface Registration)' (id=bb02b3bf-5e80-4547-842e-e60f4aa1d0ce,
url=/api/cloud/registrations/bb02b3bf-5e80-4547-842e-e60f4aa1d0ce, namespace=cloud)
2020-09-14T11:00:52.377Z audit [866bb70b-3bb5-4ac3-9a59-0b60ade7feb2] Creating a new registration for type, ui, with
client_id: ui-client-id.
```

g. If the response shows any errors, you must correct the errors.
When the logs show no errors, the saved values are updated in the management subsystem's database.

h. Confirm that the new registration exists in the registrations list by running the following toolkit CLI command:

```
apic --server server registrations:list
```

where ***server*** is the management server endpoint URL.

i. Verify that the updated registration is now in effect by running the following toolkit CLI command:

```
apic --server server registrations:get new_registration_name --output -
```

where:
- ***server*** is the management server endpoint URL
- ***new_registration_name*** is the new registration name (including any numeric suffix) following the patterns in Table 1.

j. Save the old registration (refer to Table 1 to determine the previous registration's file name) in case you need it when restoring an older backup.
Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created.

k. If you updated the registration credentials for API Designer or the toolkit, inform users so they can download new credentials.
Users can download and set up new credentials as explained in Installing the toolkit. Other credentials are updated automatically and require no actions from the user.

Configuring maximum size of client requests to the Management subsystem

You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.

About this task

The default maximum size is 8 megabytes. You can increase this value.

Increasing the setting can be useful if you get errors when using the CLI to import a large API. Example error:

```
HTTP/1.1 413 Request Entity Too Large
```

Procedure

1. Edit the Management CR. Refer to the Management CR templates:

```
spec:
  ...
  ...
  template:
  - name: juhu
    containers:
    - name: juhu
      env:
      - name: CLIENT_MAX_BODY_SIZE
        value: "12m"
```

Note that the default value is `"8m"`. The `template:` entry must be indented two spaces, and be located within the `spec:` element.

2. Apply the changes:

```
kubectl -n <namespace> apply -f management_cr.yaml
```

Setting rate limits for public APIs on the management service for a Kubernetes environment

Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.

Before you begin

Note:

- This article does not apply to the IBM® DataPower® Gateway component that is a part of the IBM API Connect solution.
- This article applies to third-party software that IBM does not control. As such, the software may change and this information may become outdated.

These instructions assume you have a working Kubernetes environment with the kubectl command-line tool installed, and that you understand how to manage Kubernetes. For more information, see https://kubernetes.io.

About this task

Rate limits can be set for public APIs on the management service. Rate limits on APIs help provide protection from DDoS (distributed denial of service) attacks. Without a rate limit, API calls from public APIs are unlimited.

The rate limit configuration requires that the header contains the actual client IP address. Any load balancer or proxy (for example, HAProxy) that is installed in front of the management service must be configured to pass the actual client IP address.

This procedure must be performed on a running API Connect deployment.

Rate limits are calculated as requests per seconds per client.

Note: If the rate limit has been reached on the management subsystem, the client will get an HTTP error: **`429 Too Many Requests`**.

Procedure

- Configure the ingress-nginx-ingress-controller to pass the real client IP address:
  1. Set the **`use-proxy-protocol`** parameter to **`true`** by entering the following command:

     ```
     kubectl edit ConfigMap -n kube-system ingress-nginx-ingress-controller
     ```

  2. In the ConfigMap, set the following: **`use-proxy-protocol: "true"`**
- Set a rate limit:
  1. You can add a **`template:`** value in your subsystem CR to provide an override.
     For example, the Management CR template value can be:

     ```
     spec:
       ...
       template:
       - name: juhu
         containers:
         - name: juhu
           env:
           - name: RATE_LIMIT_PER_CLIENT
             value: "10"
           - name: LIMIT_REQUEST_OPTION
             value: "burst=10 nodelay"
     ```

     Note that the **`template:`** entry must be indented two spaces, and must be located in the **`spec:`** element.

     In this example, the first option sets the rate to 10 requests per second (10r/s). The second option allows 10 requests boosted without delay within < 1 second. You can customize as needed.

     The **`rateLimitPerClient`** property sets **`rate`**, and **`limitRequestOption`** sets **`[burst=number] [nodelay | delay=number]`** in the following nginx configuration:

     ```
     limit_req_zone key zone=name:size rate=rate;
     limit_req zone=name [burst=number] [nodelay | delay=number];
     ```

     Note that **`zone`** has been pre-defined and can't be configured. For more details, see the nginx doc http://nginx.org/en/docs/http/ngx_http_limit_req_module.html.

  2. Apply the changes:

     ```
     kubectl -n <namespace> apply -f management_cr.yaml
     ```

  3. Validate that the juhu pod has restarted by listing the pods:

     ```
     kubectl get pods -n <namespace> | grep juhu
     ```

  4. Check the AGE column to ensure a new juhu pod has started.
  5. If the older juhu pod is still running, delete it with the following command:

     ```
     kubectl delete pods -n <namespace> <old-juhu-pod>
     ```

- Disable a rate limit:
  1. Validate that the juhu pod has restarted by listing the pods:

     ```
     kubectl get pods -n <namespace> | grep juhu
     ```

  2. Check the AGE column to ensure a new juhu pod has started.
  3. If the older juhu pod is still running, delete it with the following command:

     ```
     kubectl delete pods -n <namespace> <old-juhu-pod>
     ```

Enabling API governance on Kubernetes

You can optionally configure API governance in API Connect on a Kubernetes, OpenShift, or IBM® Cloud Pak for Integration deployment by enabling the governance microservice.

About this task

API governance is an optional add-on to IBM API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process.

Note:

- These instructions apply only to Kubernetes, OpenShift, and IBM Cloud Pak for Integration installations. For VMware installations, see Enabling API governance on VMware.
- API governance rulesets cannot be added to your deployment until the governance microservice is enabled.

To enable or disable the governance microservice, you must configure the Management subsystem custom resource (CR) file. See the following instructions:

- **Enabling the governance microservice as part of a new deployment**
- **Enabling the governance microservice as part of an existing deployment**
- **Disabling the governance microservice**

After the governance microservice is enabled, API governance resources can be created. For more information, see Configuring API governance in the Cloud Manager, and Configuring API governance in the API Manager.

Procedure

- **Enabling the governance microservice as part of a new deployment**
  Edit the CR file for the Management subsystem and add the settings for the governance microservice.

  1. Edit the `ManagementCluster` CR and add the following definition for the governance microservice. Append the governance definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.

     ```
     spec:
       ...
       governance:
         enabled: true
     ```

  2. Apply the updated CR by running the following command as part of the standard Management subsystem installation (see Installing the Management subsystem for details):

     ```
     kubectl apply -f management_cr.yaml -n <management_namespace>
     ```

     Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.
     The governance microservice will be enabled with the Management subsystem.

  3. You can monitor your Kubernetes deployments by running the following command:

     ```
     kubectl get deployments -n <management_namespace>
     ```

     The installation is complete when the `management-compliance` pods are shown in the list of returned values.
- **Enabling the governance microservice as part of an existing deployment**
  Edit the deployed CR for the Management subsystem and add the settings for the governance microservice.

  1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

     ```
     kubectl get managementcluster -n <management_namespace>
     ```

     Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.
  2. Edit the deployed CR by running the following command:

     ```
     kubectl edit managementcluster <management-cr-name> -n <management_namespace>
     ```

     Where:
     - *management-cr-name* is the name of the deployed CR for the Management subsystem.
     - *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.
  3. In the editor, append the governance definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.

     ```
     spec:
       ...
       governance:
         enabled: true
     ```

  4. Save the update.
     The governance microservice is enabled in the Management subsystem.

  5. You can monitor your Kubernetes deployments by running the following command:

     ```
     kubectl get deployments -n <management_namespace>
     ```

     The installation is complete when the `management-compliance` pods are shown in the list of returned values.
- **Disabling the governance microservice**
  Edit the deployed CR for the Management subsystem and update the settings for the governance microservice.

  1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

     ```
     kubectl get managementcluster -n <management_namespace>
     ```

     Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.
  2. Edit the deployed CR by running the following command:

     ```
     kubectl edit managementcluster <management-cr-name> -n <management_namespace>
     ```

     Where:
     - *management-cr-name* is the name of the deployed CR for the Management subsystem.
     - *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

3. In the editor, change the governance definition to `enabled: false`.

```
spec:
  ...
  governance:
    enabled: false
```

4. Save the update.
   The governance microservice is disabled in the Management subsystem.

Results

Note that when the governance microservice is enabled, there are a number of new deployments, jobs, and pods in the `ManagementCluster` namespace. These Kubernetes governance resources have names containing either `compliance-service` or `compliance-ui`. For example:

```
kubectl get pods -n apic | grep compliance
management-compliance-service-f6cdf95fc-t4qkx                     1/1     Running     0          127m
management-compliance-ui-59897fcc4-zm25v                          1/1     Running     0          126m
management-up-compliance-service-data-populate-0-to-1-t2f4d       0/1     Completed   1          132m
management-up-compliance-service-schema-0-to-1-2lkqq              0/1     Completed   0
```

Configuring monetization on Kubernetes

You can optionally configure monetization in API Connect on a Kubernetes deployment by enabling the billing microservice.

About this task

API Connect includes a subscription billing microservice that allows API providers to define pricing plans in their API Products, and monetize their API offerings. If a Product contains a pricing plan, API Consumers must enter their payment information into the Developer Portal before they can subscribe to that plan. API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Note:

- These instructions apply only to Kubernetes installations. For VMware installations, see Configuring monetization on VMware.
- To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See Firewall requirements on Kubernetes and Firewall requirements on VMware for more information about this requirement.
- Provider organizations cannot add any billing integrations until the billing microservice is enabled.
- After provider organizations have added billing integrations, the billing microservice must not be disabled.

To enable or disable the billing microservice, you must configure the Management subsystem custom resource (CR) file. See the following instructions:

- **Enabling the billing microservice as part of a new deployment**
- **Enabling the billing microservice as part of an existing deployment**
- **Disabling the billing microservice**

Procedure

- **Enabling the billing microservice as part of a new deployment**
  Edit the CR file for the Management subsystem and add the settings for the billing microservice.

  1. Edit the `ManagementCluster` CR and add the following definition for the billing microservice. Append the billing definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.

     ```
     spec:
       ...
       billing:
         enabled: true
     ```

  2. Apply the updated CR by running the following command as part of the standard Management subsystem installation (see Installing the Management subsystem for details):

     ```
     kubectl apply -f management_cr.yaml -n <management_namespace>
     ```

     Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.
     The billing microservice will be enabled with the Management subsystem.

  3. You can monitor your Kubernetes deployments by running the following command:

     ```
     kubectl get deployments -n <management_namespace>
     ```

     The installation is complete when the `management-billing` pod is shown in the list of returned values.

  After the billing microservice is enabled, provider organization owners can create billing integration resources for API providers in the API Manager UI. For more information, see Monetizing your Products.

- **Enabling the billing microservice as part of an existing deployment**
  Edit the deployed CR for the Management subsystem and add the settings for the billing microservice.

  1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

2. Edit the deployed CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

Where:
- *management-cr-name* is the name of the deployed CR for the Management subsystem.
- *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

3. In the editor, append the billing definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.

```
spec:
  ...
  billing:
    enabled: true
```

4. Save the update.
The billing microservice is enabled in the Management subsystem.

5. You can monitor your Kubernetes deployments by running the following command:

```
kubectl get deployments -n <management_namespace>
```

The installation is complete when the `management-billing` pod is shown in the list of returned values.

After the billing microservice is enabled, provider organization owners can create billing integration resources for API providers in the API Manager UI. For more information, see Monetizing your Products.

- **Disabling the billing microservice**
Important: After provider organizations have added billing integrations, the billing microservice must not be disabled.
Edit the deployed CR for the Management subsystem and update the settings for the billing microservice.

1. Retrieve the name of the deployed CR for the Management subsystem by running the following command:

```
kubectl get managementcluster -n <management_namespace>
```

Where *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

2. Edit the deployed CR by running the following command:

```
kubectl edit managementcluster <management-cr-name> -n <management_namespace>
```

Where:
- *management-cr-name* is the name of the deployed CR for the Management subsystem.
- *management_namespace* is the name of the target installation namespace in the Kubernetes cluster.

3. In the editor, change the billing definition to `enabled: false`.

```
spec:
  ...
  billing:
    enabled: false
```

4. Save the update.
The billing microservice is disabled in the Management subsystem.

Related information

- Monetizing your Products
- stripe.com

Overriding resources for Postgres components on Kubernetes V10.0.6.0

You can optionally override the CPU and Memory resources used by postgres database components.

About this task

Note: The information in this topic applies to V10.0.6.0 only.
There are three postgres components:

- `postgres`
- `pgBouncer`
- `pgBackRest`

The CPU, memory, and PVC resources used by the `postgres`, `pgBouncer`, and `pgBackRest` components depend on the deployment profile that was selected on installation. The resources that are defined for each deployment profile are shown here: Management component deployment profile limits.
Important: The postgres pod is licensed based on CPU use, so adjusting this pod's CPU limit has licensing implications, see: Component profiles, CPU limits, memory limits, and licensing

Procedure

To override resources, complete the following procedure

1. Prepare a patch yaml.
   - Ensure that you specify the name of the container for the resource you want to override. Names:
     - `postgres`
     - `pgBouncer`
     - `pgBackRest`
   - You can override multiple postgres components in parallel. Adding one container section under `pgcluster` template.
   - You can override just a single value, such as CPU limit.

   Examples:
   - Example yaml, to replace all postgres DB resources:

     ```
     spec:
       template:
       - name: pgcluster
         containers:
         - name: postgres
           resources:
             requests:
               cpu: 1500m
               memory: 8Gi
             limits:
               cpu: 600m
               memory: 512Mi
     ```

   - Example yaml, to replace all pgBouncer resources:

     ```
     spec:
       template:
       - name: pgcluster
         containers:
         - name: pgBouncer
           resources:
             requests:
               cpu: 200m
               memory: 128Mi
             limits:
               cpu: 300m
               memory: 256Mi
     ```

   - Example yaml,to replace CPU limit of `postgres` database:

     ```
     spec:
       template:
       - name: pgcluster
         containers:
         - name: postgres
           resources:
             limits:
               cpu: 1000m
     ```

   - Example yaml, to replace CPU limits of `postgres` DB and `pgBouncer`:

     ```
     spec:
       template:
       - name: pgcluster
         containers:
         - name: postgres
           resources:
             limits:
               cpu: 1000m
         - name: pgBouncer
           resources:
             limits:
               cpu: 1000m
     ```

2. Apply the patch yaml:

   ```
   kubectl patch mgmt <mgmt-cr-name> --type merge --patch "$(cat patch.yaml)"
   ```

Results

The postgres components will restart with updated values.

Overriding resources for Postgres components on OpenShift V10.0.6.0

You can optionally override the CPU and Memory resources used by postgres database components.

About this task

Note: The information in this topic applies to V10.0.6.0 only.
There are three postgres components:

- `postgres`
- `pgBouncer`
- `pgBackRest`

The CPU, memory, and PVC resources used by the `postgres`, `pgBouncer`, and `pgBackRest` components depend on the deployment profile that was selected on installation. The resources that are defined for each deployment profile are shown here: Management component deployment profile limits.
Important: The postgres pod is licensed based on CPU use, so adjusting this pod's CPU limit has licensing implications, see: Component profiles, CPU limits, memory limits, and licensing

Procedure

To override resources, complete the following procedure

1. Prepare a patch yaml.
   - Ensure that you specify the name of the container for the resource you want to override. Names:
     - `postgres`
     - `pgBouncer`
     - `pgBackRest`
   - You can override multiple postgres components in parallel. Adding one container section under `mgmt-pgcluster` template.
   - You can override just a single value, such as CPU limit.

   Examples:
   - Example yaml, to replace all postgres DB resources:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: postgres
           resources:
             requests:
               cpu: 1500m
               memory: 8Gi
             limits:
               cpu: 600m
               memory: 512Mi
     ```

   - Example yaml, to replace all `pgBouncer` resources:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: pgBouncer
           resources:
             requests:
               cpu: 200m
               memory: 128Mi
             limits:
               cpu: 300m
               memory: 256Mi
     ```

   - Example yaml,to replace CPU limit of `postgres` database:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: postgres
           resources:
             limits:
               cpu: 1000m
     ```

   - Example yaml, to replace CPU limits of `postgres` DB and `pgBouncer`:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: postgres
           resources:
             limits:
               cpu: 1000m
         - name: pgBouncer
           resources:
             limits:
               cpu: 1000m
     ```

2. Apply the patch yaml:

   ```
   oc patch apiconnectcluster <apiconnectcluster-cr-name> --type merge --patch "$(cat patch.yaml)"
   ```

Results

The postgres components will restart with updated values.

Restarting management subsystem pods

How to restart management subsystem pods in your API Connect deployment.

Procedure

1. Determine your deployment name by running the following command:

   ```
   kubectl -n <namespace> get deployment | grep <management CR>
   ```

   where
   - *<namespace>* is the namespace where the management subsystem is installed.
   - *<management CR>* is the name your management subsystem CR.
2. Restart pods by running the `kubectl` commands shown in Table 1.
   Use the deployment name that you obtained in step 1.

Table 1. Management pods and the commands to restart them

Pod	Restart command
`management-analytics-proxy`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-apim`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-client-downloads-server`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-hub`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-juhu`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-ldap`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-lur`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-natscluster`	Run for each `natscluster` pod, one at a time: `kubectl exec management-natscluster-n -- bash -c "kill -HUP 1"` A manual restart is not needed when the certificate is updated.
v10.0.6 only: `postgres-operator`	`kubectl rollout restart deployment postgres-operator -n <namespace>`
v10.0.6 only: `management-random-postgres*`	Rollout restart all the postgres deployments: `kubectl -n <namespace> get deploy -o name | grep postgres | egrep -v "backrest|pgbouncer|operator" | xargs kubectl -n <namespace> rollout restart`
v10.0.6 only: `management-random-postgres-pgbouncer`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
v10.0.7 and later: `edb-operator`	`kubectl rollout restart deployment edb-operator -n <namespace>`
`management-portal-proxy`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-taskmanager`	`kubectl rollout restart deployment/deployment_name -n <namespace>` A manual restart is not needed when the certificate is updated.
`management-tunnel`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-turnstile`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-ui`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-analytics-ui`	`kubectl rollout restart deployment/deployment_name -n <namespace>`
`management-remote-sitename-postgres`	`kubectl rollout restart deployment/deployment_name -n <namespace>`

3. Verify that all management pods are ready by running the following command:

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

   where *<namespace>* is the namespace where the management subsystem is installed.

   The restart is complete when all pods are **Running** and **Ready**.

Advanced configuration for the analytics subsystem

Specify advanced configuration for the analytics subsystem on Kubernetes, OpenShift, or Cloud Pak for Integration.

To manage advanced analytics features, see: Advanced configuration for the analytics subsystem.

Advanced configuration for the gateway subsystem

Specify advanced configuration for the gateway subsystem.

See:

- **Customizing a DataPower deployment**
  Update the `GatewayCluster` CR to configure `DataPowerService` APIs to customize your DataPower deployment.
- **Updating the gateway password, secrets, and certificates**
  How to update the gateway administrator password, the client_secret, the peering secret, and the invocation endpoint certificate.
- **Overriding the default DataPower image, version, and license**
  Update the `GatewayCluster` CR to override the image, version, and license information for the Gateway subsystem.
- **Enabling autoscaling of gateway pods**
  Enable autoscaling of gateway pods to ensure high availability of DataPower pods.
- **Configuring rate limits on the gateway nodes**
  Create an env.yaml file on each Gateway node to configure the default rate limit.
- **Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment**
  In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

Customizing a DataPower deployment

Update the `GatewayCluster` CR to configure `DataPowerService` APIs to customize your DataPower deployment.

You can set the following properties in the CR. Note that each `GatewayCluster` CR property has a corresponding `DataPowerService` API. Refer to the DataPower Operator documentation link for each property:

Table 1. Mapping of `GatewayCluster` properties to `DataPowerService` properties

GatewayCluster property	DataPowerService property	DataPowerService API Docs
`additionalDomainConfig`	`domains`	domains For domain configuration, see also: • Guides: Domain Configuration • Features: Configuration Management
`additionalExtraExe`	`extraExe`	extraExe
`additionalInitCmds`	`initCmds`	initCmds
`lifecycle`	`lifecycle`	lifecycle
`healthcheck`	`healthcheck`	healthcheck
`readinessTimeoutSeconds`	`readinessTimeoutSeconds`	readinessTimeoutSeconds
`hostAliases`	`hostAliases`	hostAliases

Examples of how to add the DataPower service properties to a `GatewayCluster` CR, in an API Connect deployment on native Kubernetes:

- Adding `additionalDomainConfig`:

```
spec:
  additionalDomainConfig:
  - name: "default"
    certs:
    - certType: "usrcerts"
      secret: "default-cert"
    - certType: "sharedcerts"
      secret: "shared-cert"
    dpApp:
      config:
      - "default-config"
      local:
      - "default-local"
```

- Extending the `apiconnect` domain:

```
spec:
  additionalDomainConfig:
  - name: "apiconnect"
    dpApp:
      config:
      - "custom-apiconnect-config"
      local:
      - "custom-apiconnect-local"
```

Updating the gateway password, secrets, and certificates

How to update the gateway administrator password, the client_secret, the peering secret, and the invocation endpoint certificate.

- **Changing the gateway administrator password**
  Change the gateway administrator password by updating the CR and reinstalling the subsystem.
- **Changing the gateway client_secret**
  The gateway subsystem receives its `client_secret` from the management subsystem.
- **Changing the secret for gateway peering**
  Update the secret that protects gateway-peering communications within a gateway cluster in your API Connect deployment.

- **Updating the gateway director and invocation endpoint profiles**
  Update the TLS profile that secures communications with the gateway subsystem in your API Connect deployment.

Changing the gateway administrator password

Change the gateway administrator password by updating the CR and reinstalling the subsystem.

About this task

Updating the administrator password for the gateway subsystem requires no measurable reduction in API enforcement performance (gateways remain "up"); however, services such as SNMP will experience a disconnect until the password is updated on all pods.

Procedure

1. Get the name of the secret that is used in the gateway CR.
   a. Run the following command to get the gateway's `secretName`:

      ```
      kubectl -n <namespace> get gatewaycluster <gw-cluster-name> -o yaml
      ```

      where *<namespace>* is the namespace where you installed the gateway subsystem, and *<gw-cluster-name>* is the name of the gateway cluster.

      In the CR, the gateway's **secretName** is in the **adminUser** section.

   b. Note down the secret name for use in a later step.
2. Scale the *<gw-cluster-name>* to 0 pods.
   a. Using the following example as a guide, set `replicaCount=0` for the *<gw-cluster-name>*.

      ```
      kubectl -n <namespace> patch gatewaycluster <gw-cluster-name> --type json -p='[{"op": "add", "path":
      "/spec/replicaCount", "value": 0 }]'
      ```

   b. Validate that `statefulset READY = 0/0`.
      Run the following command:

      ```
      kubectl  -n <namespace> get statefulset <gw-cluster-name>
      ```

      A successful response looks like the following example:

      ```
      NAME   READY   AGE
      gwv6   0/0     18h
      ```

3. Update the gateway's secret.
   a. Run the following command to edit the gateway's `secretName`:

      ```
      kubectl -n <namespace> edit secret secretName
      ```

   b. Change the value of `.data.password` to be the base64-encoded value of a new password. The new password can be any string you want but you must encode it to base64.
4. Scale the gateway back up to the correct number of pods.
   a. Using the following example as a guide, set `replicaCount=3` for *<gw-cluster-name>*.

      ```
      kubectl -n <namespace> patch gatewaycluster <gw-cluster-name> --type json -p='[{"op": "add", "path":
      "/spec/replicaCount", "value": 3 }]'
      ```

   b. Validate that `statefulset READY = 3/3` (this step might take few minutes to complete).
      Run the following command:

      ```
      kubectl  -n <namespace> get statefulset <gw-cluster-name>
      ```

      A successful response looks like the following example:

      ```
      NAME   READY   AGE
      gwv6   3/3     18h
      ```

5. Verify that the gateway subsystem is fully updated.
   Run the following command, replacing *<namespace>* with the appropriate namespace.

   ```
   kubectl -n namespace get GatewayCluster
   ```

   The update is complete when the **READY** status is **True**, and the **SUMMARY** reports that all services are online (**2/2**) for all the Gateway subsystems that were updated. Example:

   ```
   NAME   READY   SUMMARY   VERSION    RECONCILED VERSION   AGE
   gwv5   True    2/2       10.0.0.0   10.0.7.0-1219        7m31s
   gwv6   True    2/2       10.0.0.0   10.0.7.0-1219        7m32s
   ```

Changing the gateway `client_secret`

The gateway subsystem receives its **client_secret** from the management subsystem.

About this task

If you changed the **client_secret** on the management subsystem, force a dynamic reconfiguration and reregistration (DRR) as explained in How can I force re-population of the configuration data?. On completion, the gateway subsystem uses the new **client_secret** in its communications with the management subsystem.

Changing the secret for gateway peering

Update the secret that protects gateway-peering communications within a gateway cluster in your API Connect deployment.

About this task

You can update the secret for an appliance-based gateway or a Kubernetes-based gateway. This task causes a disruption in API transactions because the secret must be changed on all gateways at the same time, and all gateways must be restarted.

Procedure

1. Appliance only: Manually update the Gateway Peering Identification Credential and Validation Credential objects.
   Attention: If your gateway is not deployed on an appliance, skip to step 2.
   As soon as you begin this process, nodes will be out of sync. The nodes will remain out of sync until the entire process is complete. If the nodes do not recover from the out-of-sync condition, you might need to perform a DRR. For information about dynamic reregistration and reconfiguration and (DRR), see Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment.

2. Kubernetes only: If your gateway is deployed on Kubernetes, update the secret by completing the following steps.
   a. Determine the name of your gateway cluster by running the following command:

      ```
      kubectl -n namespace get gatewayCluster
      ```

      where ***namespace*** is the namespace where the gateway subsystem is deployed.

   b. Determine the name of the secret currently being used for gateway peering by running the following command:

      ```
      kubectl -n namespace describe gatewayCluster gateway-cluster
      ```

      where:
      - ***namespace*** is the namespace where the gateway subsystem is deployed.
      - ***gateway-cluster*** is the name that you obtained in step 2a.

      In the response, the name of the secret currently being used displays after the heading **Apic Gateway Peering TLS** as shown in the following example:

      ```
      Spec:
        Admin User:
          Secret Name:                        datapower-admin-credentials
        API Debug Probe:                      true
        API Debug Probe Expiration Minutes:   60
        API Debug Probe Max Records:          1000
        Apic Gateway Peering TLS:
          Secret Name:  gateway-peering
      ```

   c. Show the age of the secret currently being used by running the following command:

      ```
      kubectl -n namespace get secret gateway-peering-secret
      ```

      where:
      - ***namespace*** is the namespace where the gateway subsystem is deployed.
      - ***gateway-peering-secret*** is the secret name that you obtained in step 2b.

   d. Delete the secret and then immediately delete all of the gateway pods in the gateway cluster by running the following two commands:

      ```
      kubectl -n namespace delete secret gateway-peering-secret
      kubectl -n namespace delete pod gw-pod1 gw-pod2 gw-pod3
      ```

      where:

      - ***namespace*** is the namespace where the gateway subsystem is deployed.
      - ***gateway-peering-secret*** is the secret name that you obtained in step 2b.

      When you delete the secret and the pods, the following events occur automatically:
      i. The cert-manager recreates the secret.
      ii. The DataPower operator recreates the gateway pods.
      iii. Dynamic reconfiguration re-installs the API metadata onto the gateways.
      For information about dynamic reregistration and reconfiguration and (DRR), see Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment.

   e. To verify the update, show the age of the new gateway peering secret by running the following command:

      ```
      kubectl -n namespace get secret gateway-peering-secret
      ```

      where:
      - ***namespace*** is the namespace where the gateway subsystem is deployed.
      - ***gateway-peering-secret*** is the secret name that you obtained in step 2b.

      Make sure that the secret is new and does not show the same age as previously.

Updating the gateway director and invocation endpoint profiles

Update the TLS profile that secures communications with the gateway subsystem in your API Connect deployment.

Update the gateway API invocation certificate and the gateway director client certificate in the TLS profiles that are configured for your gateway service in the Cloud Manager UI. See Registering a gateway service and TLS profiles overview.

To update the gateway director server certificate on Kubernetes deployments, update the Kubernetes secret: **`gwv6-manager-endpoint/gw-gateway-manager`**.

Overriding the default DataPower image, version, and license

Update the **`GatewayCluster`** CR to override the image, version, and license information for the Gateway subsystem.

Procedure

To override these settings in the CR, add a **`spec.dataPowerOverride`** section as shown in the following example:

```
spec:
   dataPowerOverride:
     image: customregistry.com/custom-image-datapower:10.0.6.0
     version: 10.0.6.0
     license: X-XXXX-XXXXXX
```

Enabling autoscaling of gateway pods

Enable autoscaling of gateway pods to ensure high availability of DataPower pods.

Autoscaling enables the gateway deployment to scale dynamically, either vertically or horizontally.

It is not possible to use both horizontal and vertical pod scaling simultaneously. You must configure either **`HPA`** (for horizontal scaling) or **`VPA`** (for vertical scaling) as scaling method.

- To enable HPA, add the following fields to the CR:
  In OpenShift or Cloud Pak for Integration, add the fields to the **`spec.gateway`** section of the API Connect Cluster CR.

  ```
  podAutoScaling:
    hpa:
      maxReplicas: 5
      minReplicas: 3
      targetCPUUtilizationPercentage: 90
      targetMemoryUtilizationPercentage: 80
    method: HPA
  ```

- To enable VPA, add the following fields to the CR:
  In OpenShift or Cloud Pak for Integration, add the fields to the **`spec.gateway`** section of the API Connect Cluster CR.

  ```
  podAutoScaling:
    method: VPA
    vpa:
      maxAllowedCPU: 4
      maxAllowedMemory: 16G
  ```

Note that the values shown in the CR entries are just examples.

Table 1. Autoscaling configuration settings

Setting	Description
`method`	Specifies either Horizontal Pod Autoscaling or Vertical Pod Autoscaling Value must be either **`HPA`** or **`VPA`**. To disable autoscaling, leave **`method`** blank or omit the **`podAutoScaling:`** section from the CR.
`maxReplicas`	HPA. Maximum pods that can be scaled up to by the horizontal pod autoscaler. Must be a value greater than **`minReplicas`**.
`minReplicas`	HPA. Minimum pods that can be scaled down to by the horizontal pod autoscaler.
`targetCPUUtilizationPercentage:`	HPA. Percent threshold on average CPU usage by pods for triggering horizontal scaling.
`targetMemoryUtilizationPercentage`	HPA. Percent threshold on average memory usage by pods for triggering horizontal scaling.
`maxAllowedCPU`	VPA. Maximum value for CPU resource that can be assigned to a DataPower container when scaling vertically. Minimum value is set from **`DataPowerRequestsSpec.ResourceCPU`**
`maxAllowedMemory`	VPA. Maximum amount of memory that can be assigned to a DataPower container when scaling vertically. Minimum value is set from **`DataPowerRequestsSpec.ResourceMemory`**

Note:
DataPower Gateway supports long-lived connections such as such as GraphQL subscriptions or other websockets connections. These long-lived connections might not be preserved when pods are scaled down based on defined CPU or memory thresholds. Workloads with long-lived connections are more vulnerable to failed API transactions when auto-scaling is occurring.

To learn more about HPA and VPA autoscaling, see

- https://ibm.github.io/datapower-operator-doc/features/pod-auto-scaling/
- https://ibm.github.io/datapower-operator-doc/apis/datapowerservice/v1beta3#podautoscaling

Configuring rate limits on the gateway nodes

Create an env.yaml file on each Gateway node to configure the default rate limit.

The Gateway uses scale limits (instead of strict limits) by default to enforce rate limits. This improves the gateway's performance by providing the ability to handle large amounts of rate-limited traffic.

1. To configure the rate limit, create a file called env.yaml with the settings provided in Table 1:

Table 1. Rate limit settings

Setting	Type	Default	Optoins	Comments
`USE_RATE_LIMIT_DEFINITION`	boolean	`true`	• `true` Use scale limits • `false` Use strict limits	Determines the type of rate limiting to use on the Gateway node.
`RATE_LIMIT_UPDATE_MODE`	enumerated	`prior`	• `none`No limits in a plan are incremented after a limit is reached. • `all` All limits in a plan are incremented after a limit is reached. • `prior` Retains the behavior from pre-V10 releases: When the plan rate limit is reached, all limits in the plan that are sorted before that plan limit will be incremented.	Determines how the counts within a plan are incremented when a rate limit is hit. Applies only to scale limits.
`RATE_LIMIT_SORT_BY_DURATION`	boolean	`true`	• `true` Sort the rate limits by duration. • `false` Revert to the previous sort order.	Determines the sort order for rate limits in a plan, which in turn determines the sequence in which they are applied.
`RATE_LIMIT_USE_OFFSET`	boolean	`false`	• `true` Prevent all rate limit counts for an interval from being reset at the same time. • `false` Allow all rate limit counts for an interval to be reset at the same time.	Allow or prevent all of the rate limit counts for an interval from being reset at the same time. Preventing the simultaneous reset protects backends from being saturated with requests by resetting the counts for different subscribers of a plan at different points with in the interval. Applies only to scale limits.
`RATE_LIMIT_DEFINITION_RESPONSE_HEADERS`	boolean	`true`	• `true` Show the `x-rate-limit` headers. • `false` Hide the `x-rate-limit` headers.	Show or hide the `x-rate-limit` headers in an API response. Applies only to scale limits.

2. Save the file in the local:/ directory on the Gateway node so that the file persists on reboots of both physical and virtual machines. The settings apply only to the node where the file is stored, so each node in the Gateway cluster requires its own copy of the file.

Dynamically re-registering and reconfiguring a Gateway service in a Kubernetes deployment

In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

If a Gateway service is not configured properly for resiliency and is restarted, the gateways in the Gateway service will lose the configuration from the Management server. Configuration data from the Management server is maintained on the gateway service according to the gateway peering configuration on the gateways.

Preventing the loss of configuration data

For high availability in production environments, use a minimum of three gateways in the Gateway service.

Forcing re-population of the configuration data

To repopulate the configuration data, complete the steps in the following section to trigger an outage-less DRR.

Triggering an outage-less DRR

To force a Dynamic Reregistration and Reconfiguration across a peer group from within the gateway service without requiring a restart, complete the following steps using the CLI. This process flushes the primary peering object configured for the **apic-gw-service**. This task can only be performed with CLI, and requires the default admin ID for running the **diag** command.

1. Access the gateway service:
   - For a virtual gateway, run the following command to access the DataPower CLI:

     ```
     kubectl attach -it <pod_name>
     ```

   - For a physical gateway, use **ssh** to connect to the gateway service.
2. Switch to the API Connect application:

   ```
   switch <APIC_APP_DOMAIN>
   ```

3. Show the current gateway-peering-status:

   ```
   show gateway-peering-status
   ```

4. If the server that you connected to is not primary for the apic-gw-service gateway-peering object, force this server to become primary:

   ```
   config; gateway-peering-switch-primary <gateway-peering-name-for-apic-gw-service>;
   ```

5. Flush the apic-gw-service gateway-peering object:

   ```
   diag; gateway-peering-flush <gateway-peering-name-for-apic-gw-service>; exit
   ```

   When prompted, confirm the operation.

6. Disable and then re-enable the apic-gw-service object for every member of the gateway service:

   ```
   config; apic-gw-service; admin-state disabled; exit
   apic-gw-service; admin-state enabled; exit; exit
   ```

7. Confirm that the apic-gw-service was flushed and is now waiting for gateway service registration:
   Look in the debug log target for the apic-gw-service, located in logtemp:/// and check for a message similar to the following example:

   ```
   20200618T232339.332Z [0x88e000d7][apic-gw-service][notice]
   apic-gw-service(default): tid(2653): Waiting for gateway service registration.
   ```

   The DRR will be triggered by the next arrival of a webhook event. The Management server sends a heartbeat to the gateway at five-minute intervals, prompting the gateway to check whether it has lost its configuration and if so, trigger a DRR. In addition, if the Management server has previously marked a Catalog or cloud as being unavailable for a particular gateway service, a successful heartbeat triggers synchronization for that Catalog or cloud on that gateway service.

8. If you attached to a virtual gateway, you can detach by pressing Ctrl+P and then pressing Ctrl+Q.
9. (Optional) Force a BAU webhook event to be sent from API Manager to trigger the DRR:
   If you do not want to wait for the Management server to send a heartbeat, you can trigger the DRR manually by completing the following steps:

   a. Open the API Manager.
   b. Open the Catalog.
   c. Click Settings > Edit.
   d. Update the Summary field with some text so that it is modified.
   e. Click Save.

When the event is received, the DRR is initiated.

Confirm that DRR is complete by testing your APIs. You can also use the health check API.

Advanced configuration for the portal subsystem

- **Updating portal keys and secrets**
  Update the database passwords, keys, and the **client_secret** values for your portal subsystem.

Updating portal keys and secrets

Update the database passwords, keys, and the **client_secret** values for your portal subsystem.

- **Changing the secret for portal data and system tools**
  Replace the encryption key that generates the secret used for securing the database and system tools that are used by the portal subsystem.
- **Changing the portal client_secret**
  Change the **client_secret** for the portal subsystem to keep it in sync with the management subsystem.

Changing the secret for portal data and system tools

Replace the encryption key that generates the secret used for securing the database and system tools that are used by the portal subsystem.

Before you begin

If you are using a stand-alone, single data center deployment, you can update the encryption key with the PortalSecretRotation CR, as explained in Renewing the portal-ca with the PortalSecretRotation CR. The PortalSecretRotation CR is not supported for a two data center configuration or a Cloud Pak for Integration deployment, so you must update the key manually as explained in the following steps.

About this task

The portal subsystem uses a database plus several system tools to manage storage, hot backups, and cluster synchronization. The following services are all secured with the secret that is generated by the portal CR during installation:

- MySQL root account for each portal site
- Csync2 cluster synchronization service

Updating the portal-encryption-secret will make the following changes to your deployment:

- Change the root MySQL password
- Generate a new, site-specific MySQL password for each site
- Generate a new csync2 cluster key
- Re-encrypt all client_ids and client_secrets in the portal database

If you need to change the secret, the update is applied to all of the listed services. You can change the secret by generating a new value and restarting pods to pick up the new value. In a two data center deployment, you must manually copy the new secret to the other site.

Procedure

Generate a new secret and apply it to the deployment using the appropriate steps for your environment:

- Single data center
  Run the following command to generate a new secret and apply it to the deployment:

  ```
  kubectl -n namespace patch secret portal-enc-key -p "{\"data\": {\"encryption_secret\":\"$(head -c72 /dev/urandom | base64 -w0 | base64 -w0)\"}}"
  ```

  where ***namespace*** is the namespace where you installed the portal subsystem.

  The **portal-enc-key** was generated during installation and added to the CR as the **encryptionSecret** setting. When you generate a new secret, the value is updated automatically. If you saved the secret in a file after the Portal installation as recommended, be sure to update that file now.

- Two data centers
  a. Generate the new secret on DC1 by running the following command:

  ```
  head -c72 /dev/urandom | base64 -w0 | base64 -w0 > secret.txt
  ```

  The **portal-enc-key** was generated during installation and added to the CR as the **encryptionSecret** setting. When you generate a new secret, the value is updated automatically. If you saved the secret in a file after the Portal installation as recommended, be sure to update that file now.

  b. Open a terminal to each DC in the same working directory where you saved **secret.txt** (or, copy the file to somewhere you can open a terminal to each DC).
  c. In each terminal, verity that you can access **secret.txt** by typing the following command:

  ```
  echo $(<secret.txt)
  ```

  d. In each terminal, type the following command -- but do not press Enter yet:

  ```
  kubectl -n namespace patch secret portal-enc-key -p "{\"data\": {\"encryption_secret\":\"$(<secret.txt)\"}}"
  ```

  where ***namespace*** is the namespace where you installed the portal subsystem.

  e. Press Enter in both terminals at the same time (or as close in time as possible).
  There is a 5-minute time-out that starts as soon as any of the pods has the new secret mounted by Kubernetes. All pods will get the secret at slightly different times, but within about a minute of each other. If the 5-minute time-out expires before all pods across both DCs have the new secret, then the pods will rotate the new secret in at different times, resulting in errors in the logs and some down time for various pods, until they manage to rotate the new secret into use and recover.

Results

When pods restart, the new secret is used to secure the services. The **portal-www** admin logs display messages as the new secret is used to re-encrypt the database. The following example shows sample log messages when the updated secret is applied to the database:

```
[   upgrade stdout]  2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: NEW k8s encryption key detected on this system!
Reencrypting portal data
[   upgrade stderr]  2954 888ec0:73add2:7fce1b 2020-08-06 13:16:21: refresh_encryption_key: Command Started:
/opt/ibm/bin/refresh_encryption_key
FEgX8F5MXtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zhl1rVNFZTlzRnVfkGheLr9Le/Pb9jq
[   upgrade stdout]  2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: No portal service exists! Simply overriding the key
[   upgrade stdout]  2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: New key installed
(FEgX8F5MXtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zhl1rVNFZTlzRnVfkGheLr9Le/Pb9jq)
[   upgrade stderr]  2954 888ec0:73add2:7fce1b 2020-08-06 13:16:21: refresh_encryption_key: Command Finished:
/opt/ibm/bin/refresh_encryption_key
FEgX8F5MXtwvoJ8Wpxw5DB/6MKK/qPKfRt5FaId3jy5IjbpXL8wi3/33q3KOySh8/zhl1rVNFZTlzRnVfkGheLr9Le/Pb9jq with RC 0
[   upgrade stderr]  2872 888ec0:8cb2cc:73add2 2020-08-06 13:16:21: Command Finished: /opt/ibm/bin/sysupgrade with RC 0
```

If there is a portal service registered and developer portal sites installed, then this task re-encrypts that data as well. When the site encryption is complete, the log displays a "Command Finished" message for the **refresh_encryption_key** command, as shown in the following example:

```
Command Finished: /opt/ibm/bin/refresh_encryption_key <KEY> with RC 0
```

Changing the portal client_secret

Change the **client_secret** for the portal subsystem to keep it in sync with the management subsystem.

About this task

The portal subsystem uses the same **client_secret** as the management subsystem. If you change the **client_secret** on the management subsystem, update the portal subsystem by pulling the credential from the management subsystem.

Note: If you did not deploy the portal subsystem, skip this task.

Procedure

1. On a portal pod, run the following command to pull the portal service's **client_secret** from the management subsystem:

   ```
   kubectl -n namespace exec -t portal-mydc-www-0 -- bash -ic refresh_cloud_credentials
   ```

   where ***namespace*** is the namespace where you installed the portal subsystem.

2. Then, run the following command to update the credential for the portal sites:
   Note: If you did not deploy any portal sites, skip this step.

   ```
   kubectl -n namespace exec -t portal-mydc-www-0 -- bash -ic "refresh_site_credentials -a"
   ```

   where ***namespace*** is the namespace where you installed the portal subsystem.

Backing up, restoring, and disaster recovery

Backup your API Connect subsystems, configure scheduled database backups, and recover from a disaster.

Backups for the management, portal, and analytics subsystems consist of an infrastructure configuration backup and a database backup.
Note: The gateway receives all its configuration from the management subsystem, and does not require a backup.

Infrastructure configuration backup
: Your infrastructure configuration backup includes everything that you configure for API Connect in your deployment environment, such as:

  - Subsystem YAML files (or top-level CR YAML file).
  - Custom certificates. Such as certificates used to secure communication with remote systems like analytics offload targets.
  - Encryption secrets.
  - Backup secrets.

  The infrastructure configuration backup is also referred to as the "config backup".

  If you have a disaster, your config backup is essential for restoring your API Connect subsystems.

  Config backups must be taken manually, by exporting your API Connect Kubernetes artifacts to YAML files. If you make a configuration change, you must update your config backup.

Subsystem database backups
: Your subsystem database backups contain everything that API Connect users configure in the API Connect UIs, the toolkit CLI, and the REST API. Subsystem database backups for portal contain all site customizations. The analytics database backup contains all your analytics data.

  Subsystem database backups can be configured to run automatically, and can also be taken on-demand.

  Important points:

  - Subsystem database backups are for recovery in the same environment that the backups were taken, or in a different environment that has the same network configuration and form factor. If you want to move your API Connect deployment to a different environment and change the form factor, or modify your API Connect endpoints, then see Migrating from v10 to v10 on a different form factor.
  - The management and portal subsystems databases must be kept in sync with each other, so take management and portal database backups at the same time. When you perform a management or portal database restore, complete the restoration of the management database first, and then restore the developer portal.

Two data center disaster recovery deployments

Two data center disaster recovery deployments have additional backup, restore, and disaster recovery considerations. Review the Backup and restore requirements for a 2DCDR deployment before you follow the topics in this section.

- **Backing up and restoring the management subsystem**
  How to backup and restore the infrastructure configuration and database of your management subsystem.
- **Backing up and restoring the developer portal**
  How to backup and restore the infrastructure configuration and database of your portal subsystem.

- **Backing up and restoring the analytics subsystem**
  How to configure analytics backups and recover from a disaster.
- **Backing up and restoring on OpenShift top-level CR**
  How to backup and restore your top-level CR deployment of API Connect
- **Backing up and restoring on Cloud Pak for Integration and OpenShift top-level CR**
  How to backup and restore all API Connect subsystem in a Cloud Pak for Integration, or OpenShift top-level CR deployment

Backing up and restoring the management subsystem

How to backup and restore the infrastructure configuration and database of your management subsystem.

The following topics cover how to backup, restore, and recover your management subsystem from a disaster.

- **Taking a management infrastructure configuration backup**
  How to backup your management subsystem deployment configuration.
- **Configuring management database backups**
  Configure scheduled database backups of your management subsystem.
- **Running an on-demand backup of the management database**
  Deploy the `Backup` CR to initiate a backup of the management database.
- **Restoring from a management database backup**
  Restore a management database backup, either with a `ManagementRestore` custom resource (CR), or by using a backup ID.
- **Recovering from a disaster**
  Reinstall the management subsystem by using your infrastructure configuration backup, then restore the management database backup.
- **Troubleshooting management database backup and restore**
  Common problems and how to diagnose issues with backup and restore.

Taking a management infrastructure configuration backup

How to backup your management subsystem deployment configuration.

About this task

Your infrastructure configuration backup includes everything that you configure for API Connect in your deployment environment, such as:

- Subsystem YAML files (or top-level CR YAML file).
- Custom certificates.
- Encryption secrets.
- Backup secrets.

The infrastructure configuration backup is also referred to as the "config backup".

If you have a disaster, your config backup is essential for restoring your API Connect subsystems.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Export the management CR to a file called mgmt_cr_backup.yaml:

   ```
   kubectl -n <management namespace> get ManagementCluster -o yaml > mgmt_cr_backup.yaml
   ```

2. Export the management database encryption secret to a file called management_enc_key_backup.yaml:
   a. Get the name of the encryption secret:

      ```
      kubectl -n <management namespace> get mgmt -o yaml | grep encryption
      ```

   b. Take note of the `encryptionSecret` name returned:

      ```
      encryptionSecret: <management enc key>
      ```

   c. Backup the encryption secret to a YAML file:

      ```
      kubectl -n <management namespace> get secret <management enc key> -o yaml > management_enc_key_backup.yaml
      ```

3. Export your management database backup secret:

   ```
   kubectl -n <management namespace> get secret <mgmt backup secret> -o yaml > mgmt-backup-secret_backup.yaml
   ```

   The backup secret is created when you configure database backups: Create backup secret. If database backups are not yet configured, ensure that this secret is backed up when it is configured.

4. If you created a secret for your S3 store's CA certificate, take a backup of this secret:

   ```
   kubectl -n <management namespace> get secret <mgmt custom server ca> -o yaml > mgmt-custom-server-ca_backup.yaml
   ```

   The S3 CA secret is created when you configure database backups: Create S3 CA secret. If database backups are not yet configured, ensure that this secret is backed up when it is configured.
5. Export the management client application credential secrets to YAML files:
   a. Get the names of the credential secrets:

```
kubectl -n <management namespace> get mgmt -o yaml | grep CredentialSecret
```

Expected output is:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
discoveryCredentialSecret: management-discovery-cred
governanceCredentialSecret: management-governance-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

b. Backup each credential secret to a YAML file:

```
kubectl -n <management namespace> get secret <secret name> -o yaml > <secret name>_backup.yaml
```

For example, to backup the ATM credential secret:

```
kubectl -n <management namespace> get secret management-atm-cred -o yaml > management-atm-cred_backup.yaml
```

c. Edit each credential secret YAML file and remove both the **ownerReferences** subsection and the **selfLink** property.
Example of the **ownerReferences** and **selfLink** properties to be removed:

```
ownerReferences:
  - apiVersion: management.apiconnect.ibm.com/v1beta1
    blockOwnerDeletion: true
    controller: true
    kind: ManagementCluster
    name: management
    uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
  selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
```

6. Export any other certificates that you customized. For example, if you customized the **apiManagerEndpoint** certificate, save the certificate by copying the corresponding secret to a YAML file:

```
kubectl -n <management namespace> get secret apim-endpoint -o yaml > apim-endpoint_secret_backup.yaml
```

Configuring management database backups

Configure scheduled database backups of your management subsystem.

Before you begin

You must have a remote S3 object-store for storing your management database backups, for example:

- AWS S3.
- IBM Cloud Object Storage.
- Custom S3 storage.

The S3 store's backup folder must be empty.

SFTP and local backups are not supported on v10.0.7.0.

About this task

Important points:

- Management database backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a management database backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see: Migrating from v10 to v10 on a different form factor.
- Ensure that you regularly take a Taking a management infrastructure configuration backup. If a disaster occurs and you lose your management subsystem configuration, then you cannot restore your management database backup without first restoring the config backup.
- Take portal database backups at the same time as you take management database backups.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place.

Procedure

1. Create a backup secret.
The backup secret is a Kubernetes secret that contains your access key and secret for your S3 object store.
Create the backup secret with the following command:

```
kubectl -n <management namespace> create secret generic mgmt-backup-secret --from-literal=key='<access key>' \
--from-literal=keysecret='<access secret>'
```

where *<access key>* and *<access secret>* are the access credentials for your S3 store.
Take a backup of this secret and keep it with your Taking a management infrastructure configuration backup.

Important: It is recommended to regularly update your backup secret:
a. Run the same **kubectl create secret generic <secret name>** command, but with a new secret name and your updated access key and secret.

b. Edit your management CR to update the secret name in `spec.databaseBackup.credentials`.
c. Reload the secret in the management subsystem with:

```
kubectl annotate mgmt <mgmt-subsystem-name> apiconnect-operator/management-db-reloaded-at=$(date -u +"%Y-%m-%dT%H:%M:%SZ") -n <management namespace> --overwrite
```

where *<mgmt-subsystem-name>* is the name of your management subsystem CR.

2. If your S3 store has a custom CA certificate, then create a Kubernetes secret that contains this CA certificate, so that your management subsystem trusts the S3 store.
   a. Create a base64 encoded string from your S3 CA certificate file:

```
base64 <CA certificate filepate> | tr -d '\n'
```

Note: Include the complete chain in your certificate file, for example:

```
intermediate-certificate-1
intermediate-certificate-2
. . .
intermediate-certificate-n
root-certificate
```

The first certificate (intermediate-certificate-1) is the issuer of the S3 store server certificate, and each subsequent certificate is the issuer of the preceding certificate. The root certificate must be self-signed.
If the certificate includes no intermediate certificates, then the file contains only the root certificate.

   b. Create a file called mgmt_custom_s3_ca.yaml, and paste in the following text:

```
apiVersion: v1
data:
  ca.crt: <base64 encoded certificate>
kind: Secret
metadata:
  name: mgmt-custom-server-ca
type: generic
```

where *<base64 encoded certificate>* is the string that is generated from step 2.a.

   c. Create the Kubernetes secret from the YAML file:

```
kubectl -n <management namespace> apply -f mgmt_custom_s3_ca.yaml
```

   d. Keep the mgmt_custom_s3_ca.yaml file in your Taking a management infrastructure configuration backup.

3. **Kubernetes and OpenShift individual subsystem CR users:** Edit your `ManagementCluster` CR:

```
kubectl -n <management namespace> edit ManagementCluster
```

4. **Cloud Pak for Integration and OpenShift top-level CR users:** Edit your `APIConnectCluster` CR:

```
oc -n <apic namespace> edit APIConnectCluster
```

5. Configure your backup settings in the `databaseBackup` section of the `spec` object:

```
spec:
  databaseBackup:
    host: <host>/<region>
    path: <path>
    credentials: <backup secret>
    backupCerts: <custom CA certificate>
    schedule: "<cron schedule>"
    repoRetentionFull: <number of days>
    backups3URIStyle: <path/host> # host or virtual style is the default
```

Note: On Cloud Pak for Integration and OpenShift when using the top-level CR, add the `databaseBackup` section to the `spec.management` section.
Set the properties as follows:

- `host` is the endpoint of your S3 store. If your S3 endpoint does not have a region, then omit */<region>*.
- `path` is the location for your backups on the S3 store, in the format `bucket_name/folder`.
  Note:
  - The use of subdirectories in the bucket name is not supported.
  - Always use a unique path for every management subsystem deployed in your organization. S3 bucket paths must not be shared by multiple management subsystems.
  - The `path` location must be empty.
- `credentials` is the name of the backup secret that you created in step 1.
- `backupCerts` is the CA certificate secret that is created in step 2.
- `schedule` is the schedule for when database backups are run, in `cron` format:

```
    * * * * * *
    - - - - - -
    | | | | | |
    | | | | | +----- day of week (0 - 6) (Sunday=0)
    | | | | +------- month (1 - 12)
    | | | +--------- day of month (1 - 31)
    | | +----------- hour (0 - 23)
    | +------------- min (0 - 59)
    +--------------- sec (0 - 59)
```

For example:

```
0 0 1 * * *
```

runs a backup at 1 AM every day.

The default is an empty string (no backup configured).

Note: Set the same schedule as your developer portal database backups.

- `repoRetentionFull` is the number of days to retain backups for. For example, if `repoRetentionFull` is set to 30, then any backups older than 30 days are deleted.

  ```
  Minimum: 1
  Maximum: 9999999
  Default: 0 (No retention)
  ```

  Important points:
  - The `repoRetentionFull` start day is reset to the current day when you restore a management database backup.
  - `repoRetentionFull` applies to both manual backups and scheduled backups.
  - When older backups are deleted, their corresponding `backup` CR is deleted.
- `backups3URIStyle` indicates whether URIs to your S3 backup specify a host or a path value. Set to either `host` or `path`. The default is `host`. Some custom S3 providers require URI style to be set to `path`. For example, MinIO supports both host and path style setup. You can create a MinIO S3 server to accept only path style client communications.

  Important: Contact your custom S3 administrator before you configure this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the management subsystem.

6. Update the `ManagementCluster` CR backup in your Taking a management infrastructure configuration backup.
   If you are using Cloud Pak for Integration or OpenShift with the top-level CR, also update your `APIConnectCluster` CR backup.

What to do next

Run an on-demand backup to confirm that your backup configuration is correct, see Running an on-demand backup of the management database.

Running an on-demand backup of the management database

Deploy the `Backup` CR to initiate a backup of the management database.

About this task

If you are taking a management database backup, you should also take a portal database backup for all registered portal services. Management and portal databases must be kept in sync.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Verify the health of your `ManagementCluster`:

   ```
   kubectl -n <management namespace> get ManagementCluster
   ```

   The status should show `Running`, and all pods should be up:

   ```
   NAME   READY   STATUS    VERSION      RECONCILED VERSION   AGE
   mgmt   n/n     Running   10.0.7.0     10.0.7.0             19h
   ```

   Note: On Cloud Pak for Integration a `keycloak` cluster is also shown in this output. Check the status on the `ManagementCluster` only.
2. Identify the management database cluster name:

   ```
   kubectl -n <management namespace> get cluster
   ```

   Example output that shows the database cluster name is `m1-dc1-db`:

   ```
   NAME         AGE   INSTANCES   READY   STATUS                     PRIMARY
   m1-dc1-db    8d    3           3       Cluster in healthy state   m1-dc1-db-1
   ```

3. Create a file called mgmtbackup_cr.yaml and paste in the following text:

   ```
   apiVersion: postgresql.k8s.enterprisedb.io/v1
   kind: Backup
   metadata:
     generateName: mgmt-backup- # prefix for the name of the backup CR that is generated.
   spec:
     cluster:
       name: <database cluster name>
   ```

   where *<database cluster name>* is the cluster name that you identified in step 2.

4. Create the `Backup` CR from the mgmtbackup_cr.yaml file:

   ```
   kubectl -n <management namespace> create -f mgmtbackup_cr.yaml
   ```

5. Verify that the backup is running with:

   ```
   kubectl -n <management namespace> get backup -o custom-
   columns="name:.metadata.name,backupId:.status.backupId,endpoint:.status.endpointURL,path:.status.destinationPath,servernam
   e:.status.serverName,status:.status.phase"
   ```

   The expected output when the backup is completed is:

   ```
   name                  backupId          endpoint                          path                          servername
   ```

```
status
test-backup-1.17.1   20230831T110022   https://seaweed.default.svc:30333   s3://backups/backup01ibm   apic-mgmt-d54a2194-
db-2023-08-31T11:49:10+01:00   completed
```

Do not delete the **Backup** CR when the backup is complete. The **Backup** CR is required to restore the backup.

What to do next

Take a database backup of any portal subsystems that are registered with this management subsystem: Running an on-demand backup of the portal database.

Restoring from a management database backup

Restore a management database backup, either with a **ManagementRestore** custom resource (CR), or by using a backup ID.

Use a **ManagementRestore** CR when your management subsystem still has the **Backup** CR that was used to take the backup you want to restore.

Restore with a backup ID when your management subsystem does not have the **Backup** CR corresponding to the backup you want to restore. This situation can occur if the **Backup** CR was deleted, or in a disaster recovery situation where reinstall of the management subsystem is required.

- **Restoring the management database with a backup CR**
  Create a **ManagementRestore** CR that references the **Backup** CR of the backup you want to restore.
- **Restoring the management database with a backup ID**
  If the **Backup** CR is missing, but the backup is present on your S3 store, you can restore the backup by using the backup ID.

Restoring the management database with a backup CR

Create a **ManagementRestore** CR that references the **Backup** CR of the backup you want to restore.

About this task

- Management database backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a management database backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see Migrating from v10 to v10 on a different form factor.
- When you restore a management database backup, you must also restore the database backups of all registered developer portal services, so that the portal and management databases remain in sync.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place.

Procedure

1. Identify the existing backup CR that you want to restore:

   ```
   kubectl -n <management namespace> get backup -o custom-
   columns="name:.metadata.name,backupId:.status.backupId,endpoint:.status.endpointURL,path:.status.destinationPath,servernam
   e:.status.serverName,status:.status.phase"
   ```

   Example output:

   ```
   NAME                 BACKUPID          ENDPOINT                          PATH                        SERVERNAME
   STATUS
   test-backup-1.17.1   20230831T110022   https://seaweed.default.svc:30333   s3://backups/backup01ibm   apic-mgmt-d54a2194-
   db-2023-08-31T11:49:10+01:00   completed
   ```

   The backup name in this example is **test-backup-1.17.1**. You can restore the management database from any backup where the **STATUS** is **completed**.

   If you cannot find the backup CR you want, but it is present on your S3 store, then you can restore with the backup ID: Restoring the management database with a backup ID.

2. Verify that the backup is present in your S3 store:
   For example, on an AWS S3 store, run:

   ```
   aws s3 ls <path>
   ```

   where *<path>* is the location on your S3 store you configured for your management database backups.
3. Create a file called mgmtrestore_cr.yaml, and paste in the following text:

   ```
   apiVersion: management.apiconnect.ibm.com/v1beta1
   kind: ManagementRestore
   metadata:
     generateName: management- # this the prefix for the name of the restore CR that is created
   spec:
     backupName: <backup CR name>
     subsystemName: <management CR name>
   ```

   where
   - *<backup CR name>* is the backup name that you identified in step 1.
   - *<management CR name>* is the name of your **ManagementCluster** CR, run **kubectl get mgmt** to see what this name is.

Note: If the management database encryption key was changed since the backup was taken (because you followed the steps in Changing the database encryption key), then you must restore the original encryption key before you apply the restore CR. See restore old encryption key.

4. Start the restore by creating the **ManagementRestore** CR from the YAML file:

```
kubectl -n <management namespace> create -f mgmtrestore_cr.yaml
```

5. Verify that the restore is running with:

```
kubectl -n <management namespace> get ManagementRestore --sort-by=.metadata.creationTimestamp
```

Example output:

```
NAME                  STATUS      BACKUP              CLUSTER       MESSAGE                                          PITR
AGE
mgmt-restore-hr9zx    Complete    mgmt-backup-xnjkl   m1            Restore process completed (DB Restore + DRR job)   9m8s
```

What to do next

Restore the database backups of any developer portal services that are registered with the management subsystem: Restoring from a portal database backup.

Restoring the management database with a backup ID

If the **Backup** CR is missing, but the backup is present on your S3 store, you can restore the backup by using the backup ID.

About this task

Restore of the management database with a backup ID is required in a disaster recovery scenario, where you redeployed your management subsystem and restored your infrastructure configuration from your Taking a management infrastructure configuration backup.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place.

Procedure

1. Identify the backup ID and path of the backup that you want to restore from your S3 store.
In your S3 store, management database backups appear as folders that use a datestamp for their name, and are located inside the backup path. Example of management subsystem and its full path on an S3 store:

```
apic-mgmt-backups/impr-restore/m1-65288d1d-db-2023-09-14T01:30:40Z/base/20230914T013040
```

   - **apic-mgmt-backups/impr-restore/m1-65288d1d-db-2023-09-14T01:30:40Z** is the backup path. The datestamp **2023-09-14T01:30:40Z** indicates when the management database was created (not when the backup was created). The management database is created on initial installation, and recreated when a backup restore is run. The path with the most recent datestamp should contain the most recent backups.
   - **20230914T013040** is the backup ID. The format indicates the date and time that the backup was taken.
   - **base** is an autogenerated path segment and is ignored.

2. Create a file called mgmt-restore.yaml and paste in the following text:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management- # this the prefix for the name of the restore CR that is created
spec:
  subsystemName: <management CR name>
  backup:
    host: <host>/<region>
    path: <path>
    credentials: <backup secret>
  backupId: <backup ID>
```

where:
   - **subsystemName** is the name of your **ManagementCluster** CR, run **kubectl get mgmt** to see what this name is.
   - **host** is the endpoint of your S3 store. If your S3 endpoint does not have a region, then omit */<region>*.
   - **path** is the path that you identified in step 1.
   - **credentials** is the backup secret that you configured to authenticate with the S3 store. See Create backup secret.
   - **backupId** is the ID that you identified in step 1.

Example mgmt-restore.yaml file that uses the values from step 1:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management-
spec:
  subsystemName: management
  backup:
    host: s3.ibm.com/us-west
    path: pic-mgmt-backups/impr-restore/m1-65288d1d-db-2023-09-14T01:30:40Z
    credentials: mgmt-backup-secret
  backupId: 20230914T013040
```

Note: If the management database encryption key was changed since the backup was taken (because you followed the steps in Changing the database encryption key), then you must restore the original encryption key before you apply the restore CR. See restore old encryption key.

3. Start the restore by creating the **ManagementRestore** CR from the mgmt-restore.yaml file:

```
kubectl -n <management namespace> create -f mgmt-restore.yaml
```

4. Verify that the restore is running with:

```
kubectl -n <management namespace> get ManagementRestore --sort-by=.metadata.creationTimestamp
```

Example output:

```
NAME                 STATUS      BACKUP              CLUSTER       MESSAGE                                          PITR
AGE
mgmt-restore-hr9zx   Complete    mgmt-backup-xnjkl   m1            Restore process completed (DB Restore + DRR job)   9m8s
```

What to do next

Restore the database backups of any developer portal services that are registered with the management subsystem: Restoring from a portal database backup.

Recovering from a disaster

Reinstall the management subsystem by using your infrastructure configuration backup, then restore the management database backup.

Before you begin

To successfully recover the management subsystem, you must have the following backups:

- Management infrastructure configuration backup.
- Management database backup

About this task

- Management subsystem backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a management subsystem backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see Migrating from v10 to v10 on a different form factor.
- When you restore a management database backup, you must also restore the database backups of all registered developer portal services, so that the portal and management databases remain in sync.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Reinstall your management subsystem by using the YAML files from your infrastructure configuration backup.
   Your infrastructure configuration backup consists of:
   - Management CR YAML: mgmt_cr_backup.yaml.
   - Management database encryption secret: management_enc_key_backup.yaml.
   - Management backup secret: mgmt-backup-secret_backup.yaml
   - Multiple <credential secret>_backup.yaml files, one for each credential secret.
   - If you customized any certificates, a YAML file of the secret for each customized certificate. For example: apim-endpoint_secret_backup.yaml.

   a. Create the database encryption secret with the management_enc_key_backup.yaml file:

   ```
   kubectl -n <management namespace> apply -f management_enc_key_backup.yaml
   ```

   b. Create the backup credential secret with your mgmt-backup-secret_backup.yaml file:

   ```
   kubectl -n <management namespace> apply -f mgmt-backup-secret_backup.yaml
   ```

   c. Create the credential secrets with the <credential secret>_backup.yaml files you saved in step 5 of Taking a management infrastructure configuration backup:

   ```
   kubectl -n <management namespace> apply -f <credential secret backup file>
   ```

   d. Set the S3 CA secret (if required) from your backup file:

   ```
   kubectl -n <management namespace> apply -f mgmt_custom_s3_ca.yaml
   ```

   e. Create the secrets for any customized certificates from their backup files:

   ```
   kubectl -n <management namespace> apply -f <custom certificate secret backup file>
   ```

   f. Install the management subsystem. Follow Installing the Management subsystem, but use the settings from the `spec` section of your mgmt_cr_backup.yaml file.
2. Restore your management database: Restoring the management database with a backup ID.

What to do next

Restore the backups of any portal subsystems that are registered with the management subsystem. See Backing up and restoring the developer portal.

Troubleshooting management database backup and restore

Common problems and how to diagnose issues with backup and restore.

Backup or restore CR stuck in running or in a failed state.

Describe the backup or restore CR for more information on the problem:

```
kubectl -n <namespace> describe backup backup-1700143802

Name:         management-1700143800
Namespace:    apic
Labels:       k8s.enterprisedb.io/cluster=management-dc1-db
              k8s.enterprisedb.io/immediateBackup=false
              k8s.enterprisedb.io/scheduled-backup=management
Annotations:  <none>
API Version:  postgresql.k8s.enterprisedb.io/v1
Kind:         Backup
Metadata:
  Creation Timestamp:  2023-11-16T14:10:00Z
  Generation:          1
  Resource Version:    2316802
  UID:                 c7a72883-1f6f-44a7-b360-c18f8c3ba24d
Spec:
  Cluster:
    Name:  management-dc1-db
Status:
  Backup Id:          20231116T141005
  Backup Name:        backup-1700143802
  Begin LSN:          0/D000028
  Begin Wal:          000000010000000000000D
  Destination Path:   s3://ent-edb-bnr/2dcdr-mgmt-active-1538
  End LSN:            0/D01AB58
  End Wal:            000000010000000000000D
  Instance ID:
    Container ID:  cri-o://13f5f44d0b84cb18b8a88b0f747f4c95f24a5e2c1423f5ec0a8615463874ffa2
    Pod Name:      management-dc1-db-1
  Phase:            completed
  s3Credentials:
    Access Key Id:
      Key:   key
      Name:  mgmt-backup-secret
    Region:
      Key:   region
      Name:  mgmt-backup-secret
    Secret Access Key:
      Key:       keysecret
      Name:      mgmt-backup-secret
  Server Name:  management-dc1-db-2023-11-16T10:56:39Z
  Started At:   2023-11-16T14:10:05Z
  Stopped At:   2023-11-16T14:10:29Z
Events:
  Type    Reason     Age   From                           Message
  ----    ------     ----  ----                           -------
  Normal  Starting   14m   cloud-native-postgresql-backup  Starting backup for cluster management-dc1-db
  Normal  Starting   14m   instance-manager                Backup started
  Normal  Completed  13m   instance-manager                Backup completed
```

Unable to find database backup CRs

If **kubectl get backup** does not return the backup that you want to restore, then search for the backup on your S3 object-store.
The path to API Connect management database backups on an S3 object-store has the following format:

<backup path>/<mgmt db cluster name>-<time when db was created>/base/<backup ID>

- *<backup path>* is the path that is defined in your management CR: backup settings. The format is **bucket_name/folder**.
  If your API Connect deployment was upgraded from v10.0.6.0 or earlier, then **/edb** is appended to the path.
- *<mgmt db cluster name>* is the name of your management database cluster. Identify this name with:

  kubectl -n *<management namespace>* get cluster -n active

  See get cluster name.
- *<time when db was created>* is a timestamp of when the management subsystem database was created. The format is: **2023-12-25T00:00:00Z**.
  Whenever you do a management database restore, a new management database is created, and so a new directory is created where subsequent database backups are stored.
- *<backup ID>* is the ID of a management database backup. The format is **YYYYMMDDTHHMMSS**. This is a directory that contains all the files that comprise the database backup.

Example path:

s3bucket1/folder2/mgmt-db-2023-12-25T00:00:00Z/base/20231225T094400

With the information identified in the S3 object-store, you can restore the backup by using the Restoring the management database with a backup ID method.

Common problems with backup and restore

These are some frequent issues that can cause backup or restore to fail:

- Invalid login credentials for your remote backup server. Check that the username and password that you configured in your subsystem database backup settings is correct.
- The user does not have write permissions on the backup server. Check that the username you configured in your subsystem database backup settings has write permission to the specified backup path.
- Remote backup server storage full. Check your remote backup server and if the storage is full either extend the storage space or delete older backups.

- No network access to the remote backup server. Check that you can communicate with your remote backup server from your API Connect environment.
- TLS handshaking failure with the remote backup server. If your remote backup server has a self-signed CA certificate, check that this certificate is trusted by your API Connect deployment, see the database backup configuration steps for your subsystem for more information.

Backing up and restoring the developer portal

How to backup and restore the infrastructure configuration and database of your portal subsystem.

- **Taking a portal infrastructure configuration backup**
  Create YAML backup files of your portal subsystem's Kubernetes configuration.
- **Configuring portal database backups**
  Configure scheduled database backups of your portal subsystem.
- **Running an on-demand backup of the portal database**
  Deploy the `PortalBackup` CR to initiate a backup of the portal database.
- **Managing your remote portal database backups**
  How to manage your remote portal database backup resources.
- **Restoring from a portal database backup**
  Create a `PortalRestore` CR to restore your management database.
- **Recovering from a disaster**
  Reinstall the portal subsystem by using your infrastructure configuration backup, then restore the portal database backup.
- **Troubleshooting portal database backup and restore**
  Troubleshoot problems with portal database backups and restores.

Taking a portal infrastructure configuration backup

Create YAML backup files of your portal subsystem's Kubernetes configuration.

About this task

A portal infrastructure configuration backup is essential for disaster recovery. Take an infrastructure configuration backup after installation, and whenever you make an update to your portal's Kubernetes configuration.

If you have multiple portal clusters in the same namespace, include the portal CR name in all `kubectl get PortalCluster` commands provided in this topic. For example:

```
kubectl -n <portal namespace> get PortalCluster <portal CR name>
```

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Export the portal CR to a file called portal_cr_backup.yaml:

   ```
   kubectl -n <portal namespace> get PortalCluster -o yaml > portal_cr_backup.yaml
   ```

2. Export the portal database encryption secret to a file called portal_enc_key_backup.yaml:
   a. Get the name of the encryption secret:

      ```
      kubectl -n <portal namespace> get ptl -o yaml | grep encryption
      ```

   b. Take note of the `encryptionSecret` name returned:

      ```
      encryptionSecret: <portal enc key>
      ```

   c. Backup the encryption secret to a YAML file:

      ```
      kubectl -n <portal namespace> get secret <portal enc key> -o yaml > portal_enc_key_backup.yaml
      ```

3. Export your portal database backup secret:

   ```
   kubectl -n <portal namespace> get secret <portal backup secret> -o yaml > ptl-backup-secret_backup.yaml
   ```

   The backup secret is created when you configure portal database backups. If database backups are not yet configured, remember to backup this secret when they are configured.
4. If used, export your remote portal backup server's CA secret:

   ```
   kubectl -n <portal namespace> get secret <portal custom server ca> -o yaml > portal-custom-server-ca_backup.yaml
   ```

   The S3 CA secret is created when you configure database backups: Create backup server CA secret. If database backups are not yet configured, remember to backup this secret when they are configured.
5. Export any other certificates that you customized. For example, if you customized the `portalWeb` endpoint certificate, save the certificate by copying the corresponding secret to a YAML file:

   ```
   kubectl -n <portal namespace> get secret <portalWeb secret> -o yaml > portalWeb_secret_backup.yaml
   ```

What to do next

Configure scheduled database portal backups: Configuring portal database backups.

Important: If you update your portal subsystem's Kubernetes configuration, remember to update your infrastructure configuration backup.

Configuring portal database backups

Configure scheduled database backups of your portal subsystem.

About this task

The developer portal by default takes database backups of itself every 24 hours and stores the last three days of backups in the portal subsystem PVCs (older backups are automatically deleted). These backups are called local backups because they are stored locally rather than on remote backup servers. For disaster recovery, you must configure the portal to create and store backups on a remote server.

Supported remote servers for backup storage are:

- S3 object-store, for example:
  - AWS S3.
  - IBM Cloud Object Storage.
  - Custom S3 storage.

  Only password-based authentication is supported for S3, not authentication based on public certificates and private keys. Password-based authentication for S3 requires that you generate an access key and secret.
- SFTP
  Supported credentials types:
  - Username and password.
  - Username and SSH-key. Only OpenSSH keys are supported.[1]

Important points:

- Portal database backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a portal database backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see: Migrating from v10 to v10 on a different form factor.
- Keep your portal's Infrastructure configuration backup up to date with any changes in your portals Kubernetes configuration. If a disaster occurs and you lose your portal subsystem configuration, then you cannot restore your portal database backup without first restoring the config backup.
- Take management database backups at the same time as you take portal database backups. One backup per day is recommended.
- The portal keeps a log of the last 30 days of remote backups that are taken, configured with the `portalBackupRecordDays` property. Configure the retention policy on your remote backup server to match or exceed the `portalBackupRecordDays` value.
- To disable scheduled database backups, you must remove the entire `databaseBackup` section from the portal CR. Local backups continue to be taken in all cases.
-

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Create your backup secret:
   - For SFTP backup destination, use:
     - Username and password credentials:

       ```
       kubectl create secret generic portal-backup-secret --from-literal=username='<username>'
       --from-literal=password='<password>' -n <portal namespace>
       ```

     - Username and SSH-key credentials:

       ```
       kubectl create secret generic portal-backup-secret --from-literal=username='<username>'
        --from-file=ssh-privatekey='<private key filepath>' -n <portal namespace>
       ```

   - For S3 backup destination, use:

     ```
     kubectl create secret generic portal-backup-secret --from-literal=username='<S3 access key or username>'
         --from-literal=password='<S3 access key secret or password>' -n <portal namespace>
     ```

   Keep the backup secret in your Infrastructure configuration backup.
2. **S3 storage only:** If your S3 store has a custom CA certificate, then create a Kubernetes secret that contains this CA certificate, so that your portal subsystem trusts the S3 store.
   a. Create a base64 encoded string from your S3 CA certificate file:

   ```
   base64 <CA certificate filepate> | tr -d '\n'
   ```

   Note: Include the complete chain in your certificate file, for example:

   ```
   intermediate-certificate-1
   intermediate-certificate-2
   . . .
   intermediate-certificate-n
   root-certificate
   ```

   where the first certificate (intermediate-certificate-1) is the issuer of the S3 store server certificate, and each subsequent certificate is the issuer of the preceding certificate. The root certificate must be self-signed.
   If the certificate includes no intermediate certificates, then the file contains only the root certificate.

   b. Create a file called portal_custom_s3_ca.yaml, and paste in the following text:

```
apiVersion: v1
data:
  ca.crt: <base64 encoded certificate>
kind: Secret
metadata:
  name: portal-custom-server-ca
type: generic
```

where *<base64 encoded certificate>* is the string that is generated from step 2.a.

c. Create the Kubernetes secret from the YAML file:

```
kubectl -n <portal namespace> apply -f portal_custom_s3_ca.yaml
```

d. Keep the portal_custom_s3_ca.yaml file in your Infrastructure configuration backup.

3. **Kubernetes and OpenShift inidividual subsystem CR users:** Edit your `PortalCluster` CR:

```
kubectl -n <portal namespace> edit PortalCluster
```

4. **Cloud Pak for Integration and OpenShift top-level CR users:** Edit your `APIConnectCluster` CR:

```
oc -n <apic namespace> edit APIConnectCluster
```

5. Configure your backup settings in the `portalBackup` section of the `spec` object:
   - For S3 object-store backup destination, use:

```
spec:
  portalBackup:
    credentials: <portal backup secret>
    host: <host>/<region>
    path: <path>
    port: <port>
    protocol: objstore
    backupCerts: <custom CA certificate>
    backups3URIStyle: <path/host>
    schedule: '<cron schedule>'
    portalBackupRecordDays: 30
```

   - For SFTP backup destination, use:

```
spec:
  portalBackup:
    credentials: <portal backup secret>
    host: <sftp server hostname>
    path: <directory path>
    port: <port>
    protocol: sftp
    schedule: '<cron schedule>'
    portalBackupRecordDays: 30
```

Note: On Cloud Pak for Integration and OpenShift when using the top-level CR, add the `databaseBackup` section to the `spec.portal` section.
Set the properties as follows:

Table 1. Developer Portal backup settings

Setting	Description
`credentials`	The name of the secret you created in step 1.
`host`	The hostname or your remote backup storage target: • S3 - `host`: is the S3 endpoint and region (if your S3 endpoint has a region) in the format `<S3endpoint>` or `<S3endpoint>/<S3region>`. • SFTP - the hostname of the SFTP server

`path`	The location for the backups on your remote storage target:
	• S3 - the name of your S3 bucket and path. Valid examples:
	`bucket`
	`bucket/path`
	`bucket/lots/of/paths`
	• SFTP - the full absolute path of the folder on the SFTP server, beginning with `/`.

`port`	The port on the remote storage target. Defaults are: • S3 - 443. • SFTP - 22.
`protocol`	The backup type. Specify one of the following values: • `sftp` - For backups to a remote SFTP server. • `objstore` - For backups to a remote S3 object-store. • `local` - To store backups locally, in the portal subsystem's PVCs. Note: If a disaster occurs your portal PVCs might be lost or corrupted, so it is recommended to use one of the remote backup server options.
`backupCerts`	**S3 storage only:** The secret that contains your S3 store's CA certificate, that you create in step: 2.
`backups3URIStyle`	**S3 storage only:** `backups3URIStyle` indicates whether URIs to your S3 backup should specify a host or a path value. Default is `host`. Valid values: `host` and `path`. Some custom S3 providers require URI style to be set to `path`. For example, MinIO supports both `host` and `path` style setup. You can create a MinIO S3 server to accept only `path` style client communications. Important: Contact your custom S3 administrator before you configure this field. If not properly configured, an upstream custom S3 object-store can reject connections.

Setting	Description

`schedule`	The schedule for how often automatic Developer Portal backups are run. The format for the schedule is any valid cron string. The time zone for backups is Coordinated Universal Time.
	Valid for all backup types: S3, SFTP, and local.
	
	`spec:`
	`  portalBackup:`
	`  ...`
	`  schedule: '0 2 * * *'`
	
	If the schedule property is omitted, the default schedule of one backup per day is implemented.

`portalBackupRecordDays`	The `portalBackupRecordDays` setting denotes the number of days that backup records (backups that exist on the remote backup server) are shown when you list your remote backups. It is defaulted to 30 days.

6. Update the `PortalCluster` CR backup in your Infrastructure configuration backup.
   If you are using Cloud Pak for Integration or OpenShift with the top-level CR, also update your `APIConnectCluster` CR backup.

What to do next

Run an on-demand backup to confirm that your backup configuration is correct, see Running an on-demand backup of the portal database.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Running an on-demand backup of the portal database

Deploy the `PortalBackup` CR to initiate a backup of the portal database.

About this task

Note: If you are taking a portal database backup, you must also take a management database backup at the same time. Management and portal databases must be kept in sync.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Decide which type of backup you want to take:
   - `all`: backs up the Developer Portal system and all its sites.
   - `system`: backs up the Developer Portal system only.
   - `site`: backs up a specific Developer Portal site.

   Note: Developer Portal system backup includes all the portal configuration that is not specific to sites, such as IP allowlist.
2. Identify the name of the portal cluster that you want to backup:

   ```
   kubectl -n <portal namespace> get PortalCluster
   ```

3. Create a file called portalbackup_cr.yaml and paste in the following text:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalBackup
   metadata:
     generateName: portal-bup-
   spec:
     type: <type>
     portalCluster: <portal cluster name>
     crType: create
     siteName: <site to backup>
     comment: "<comment>"
   ```

   where:
   - `type` is the backup type that you decided in step 1.
   - `portalCluster` is the cluster name that you identified in step 2.
   - `siteName` is the name of the portal site that you want to backup. This property applies only when backup type of `site` is specified. Valid settings are:
     - Site UUID or URL - back up the specified site only.
     - `installed` - backups all installed sites.
   - `comment` is a text comment to help you identify the reason for the backup, for example: "pre-upgrade backup".
4. Create the `PortalBackup` CR from the portalbackup_cr.yaml file:

   ```
   kubectl -n <portal namespace> create -f portalbackup_cr.yaml
   ```

5. Verify that the backup is running with:

   ```
   $ kubectl -n <portal namespace> get portalbackup --sort-by=.status.backupId
   ```

   The expected output for an `all` type backup is:

   ```
   NAME                    ID               TASK ID  STATUS     TYPE     CR TYPE   CLUSTER    AGE     COMMENT
   portal-record-89dgz     20200526.142449  n/a      Ready      system   record    portal     55m
   portal-record-89dgz     20200526.142458  n/a      Ready      site     record    portal     54m
   p1-bup-st4l2                             n/a      Ready      all      create    portal     56m     y
   ```

   For every on-demand backup triggered at least two `PortalBackup` CRs are created:

- **Create**. The **create** CR type is the first CR to be created. The **create** CR does not create the portal database backup itself, but is responsible for creating **record** type **PortalBackup** CRs, which are representations of individual backups.
- **Record**. The **record** **PortalBackup** CR has two types:
  - **System**. The **System** record type represents one system backup file, containing the portal configuration that applies to all sites
  - **Site**. The **Site** record type represents a backup file for one site. Site backups contain any Drupal customizations made to the site.

6. You can get a detailed view of a specific backup CR by using the command:

```
kubectl -n <portal namespace> get portalbackup <backup name> -o yaml
```

Example output for a **create** type backup CR:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-
standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnnjnu
  message: ""
  phase: Ready
  state: ""
```

Example **record** CR type backup:

```
Name:         portal-record-gxb4s
Namespace:    default
Labels:       createdBy=ibm-apiconnect
Annotations:  <none>
API Version:  portal.apiconnect.ibm.com/v1beta1
Kind:         PortalBackup
Metadata:
  Creation Timestamp:  2023-09-22T08:38:01Z
  Generate Name:       portal-record-
  Generation:          1
  Owner References:
    API Version:           portal.apiconnect.ibm.com/v1beta1
    Block Owner Deletion:  true
    Controller:            true
    Kind:                  PortalCluster
    Name:                  portal
    UID:                   1318ae60-20f5-457a-aa86-7cf3961e9bf3
  Resource Version:        640709
  UID:                     7fd18137-c77d-440f-8393-5699f6b22457
Spec:
  Comment:
  Cr Type:         record
  Portal Cluster:  portal
  Site Name:
  Type:            system
Status:
  Backup Id:     20230922.083722
  Comment Left:
```

```
  Conditions:
    Last Transition Time:  2023-09-22T08:38:02Z
    Message:
    Reason:                BackupStatusUnknown
    Status:                False
    Type:                  Warning
    Last Transition Time:  2023-09-22T08:38:02Z
    Message:               Record Backup CR for _portal_system_backup-20230922.083722.tar.gz
    Reason:                BackupComplete
    Status:                True
    Type:                  Ready
    Last Transition Time:  2023-09-22T08:38:02Z
    Message:
    Reason:                BackupInProgress
    Status:                False
    Type:                  Running
    Last Transition Time:  2023-09-22T08:38:02Z
    Message:
    Reason:                BackupFailed
    Status:                False
    Type:                  Error
  Full Backup Name:        _portal_system_backup-20230922.083722.tar.gz
  Id:                      n/a
  Message:
  Phase:                   Ready
  State:
Events:                    <none>
```

Observe the property showing the filename of the backup: portal_system_backup-20230922.083722.tar.gz.

What to do next

Take a database backup of the management subsystem that your portal is registered with. You must have management and portal database backups that are in sync with each other: Running an on-demand backup of the management database.

Managing your remote portal database backups

How to manage your remote portal database backup resources.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place.
When you run the command **kubectl get portalbackup -o wide**, you see a list of all the backups that ran during the past 30 days:

```
NAME                  ID                TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE   COMMENT   FILENAME
my-backup-f91ms       n/a               20b47fe090f5 Ready    all      create    portal    58m
portal-record-2vcgs   20231105.181212   n/a          Ready    system   record    portal    55m
_portal_system_backup-20231105.181212.tar.gz
portal-record-9j81t   20231104.095949   n/a          Ready    system   record    portal    55m
_portal_system_backup-20231104.095949.tar.gz
portal-record-c4d5s   20231106.121946   n/a          Ready    site     record    portal    25m             portal.fyre-ci-
162245-master.fyre.ibm.com@ibm@api-connect-catalog-2-20231106.121946.tar.gz
portal-record-c7p5q   20231106.110330   n/a          Ready    system   record    portal    55m
_portal_system_backup-20231106.110330.tar.gz
portal-record-f5h2h   20231106.115701   n/a          Ready    system   record    portal    55m
_portal_system_backup-20231106.115701.tar.gz
```

PortalBackup CRs with **CR TYPE** of **create** can be ignored. Create type CRs are generated when a backup is triggered manually, they are used to generate the **record** type **PortalBackup** CRs that hold the actual portal database backup. You can delete **create** type CRs after its corresponding **record** type CRs are created.
You can filter out **create** type CRs from **kubectl get portalbackup** output with **grep**:

```
kubectl get portalbackup | grep record
```

When you create a backup of type 'all', the following **PortalBackup** CRs are created:

- **PortalBackup** CR of type 'system' that contains the portal system data, which is everything in the portal database that applies to all sites.
- For each site, a **PortalBackup** CR of type 'site' that contains all the data that is related to the site, such as Drupal site customizations.

For example if your portal has three sites, you get 4 **PortalBackup** CRs:

```
portal-record-222cg   20230108.010033                Ready    system   record
portal-record-22xdc   20230108.010035                Ready    site     record
portal-record-245j7   20230108.010037                Ready    site     record
portal-record-246hm   20230108.010040                Ready    site     record
```

The portalBackupRecordDays property defines how many **PortalBackup** CRs are shown in the output of **kubectl get portalbackup**. The default is the last 30 days of CRs.

Portal performance can be negatively impacted if you have too many **PortalBackup** CRs, particularly during an upgrade. If you are taking backups more frequently than once a day, or have many sites, then it is recommended to reduce the **portalBackupRecordDays** property to a lower value. You can still keep as many backups as you want on your remote server, but if they don't have a corresponding **PortalBackup** CR then you must restore them using the .tgz file name or **timestamp**, see: Restoring from a portal database backup.

Portal backup .tgz files

For most operations that involve portal database backups, you work with a backup ID or a datestamp of the backup you want to restore, but some operations might require identifying the backup file name.

Portal database backups are contained in compressed archived files (TGZ), the name of the file is shown in the last column of the `kubectl get portalbackup -o wide` output:

```
kubectl -n <portal namespace> get portalbackup -o wide
```

Example output:

```
NAME                   ID                TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE   COMMENT   FILENAME
portal-record-2vcgs    20231105.181212   n/a          Ready    system   record    portal    55m
_portal_system_backup-20231105.181212.tar.gz
```

Restoring from a portal database backup

Create a `PortalRestore` CR to restore your management database.

Before you begin

Restore the management database backup on your management subsystem first. Restore from a management database backup that was taken at the same time as the portal database backup you want to restore.

About this task

- Portal database backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a portal database backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see Migrating from v10 to v10 on a different form factor.
- Two data center disaster recovery deployments have extra backup and restore considerations. Review Backup and restore requirements for a 2DCDR deployment before you follow the instructions in this topic.

Three types of restore are available:

- `all`: restores the Developer Portal system and all the sites that are contained in the specified database backup.
- `system`: restores the Developer Portal system but not any sites from the database backup.
- `site`: restores a specific Developer Portal site from the database backup.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Identify the portal backup that you want to restore with:

   ```
   $ kubectl -n <portal namespace> get portalbackup -o wide --sort-by=.status.backupId | grep record
   ```

   Example output:

   ```
   NAME                   ID                TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE   COMMENT   FILENAME
   portal-record-2vcgs    20231105.181212   n/a          Ready    system   record    portal    55m
   _portal_system_backup-20231105.181212.tar.gz
   portal-record-9j81t    20231104.095949   n/a          Ready    system   record    portal    55m
   _portal_system_backup-20231104.095949.tar.gz
   portal-record-c4d5s    20231106.121946   n/a          Ready    site     record    portal    25m             portal.fyre-ci-
   162245-master.fyre.ibm.com@ibm@api-connect-catalog-2-20231106.121946.tar.gz
   portal-record-c7p5q    20231106.110330   n/a          Ready    system   record    portal    55m
   _portal_system_backup-20231106.110330.tar.gz
   portal-record-f5h2h    20231106.115701   n/a          Ready    system   record    portal    55m
   _portal_system_backup-20231106.115701.tar.gz
   ```

   Note: You can restore the portal database only from backups where the `CR TYPE` is `record`.
   You can get more information on a specific backup with:

   ```
   kubectl -n <portal namespace> get portalbackup <backup name> -o yaml
   ```

2. Verify that the backup you want to restore exists on the remote server.
3. Identify the name of the portal cluster that you want to restore to:

   ```
   kubectl -n <portal namespace> get PortalCluster
   ```

4. Create a file called portalrestore_cr.yaml. The text that you paste into the YAML file depends on the type of backup you want to restore:
   - Restore type `all`. Restore the full portal database, including all sites:

     ```
     apiVersion: portal.apiconnect.ibm.com/v1beta1
     kind: PortalRestore
     metadata:
       generateName: portal-restore-
     spec:
       type: all
       portalCluster: <portal cluster name>
       dryRun: <true or false>
       timestamp: <datestamp>
       priorityList:
         - <most important site to restore>
         - <second most important site to restore>
     ```

where:

- `portalCluster` is the cluster name that you identified in step 3.
- `dryRun` if you want to test restore of the backup, without overwriting your portal, set `dryRun` to `true`.
- `timestamp` specify a timestamp to retrieve the backup from. The nearest backup, searching backwards from this timestamp, is used. Format is *'YYYYMMDD.HHMMSS'*, or set to `now` to restore the most recent backup.
- `priorityList` Prioritizes the restore of specific site. The sites are restored in the order in which they are listed in `priorityList`, followed by any remaining unlisted sites. Example contents:

  ```
  priorityList:
      - production.site.com
      - test.site.com
  ```

  If you have no preference for site restore priority, then omit this property.

- Restore type `site`. Restore backup of a particular site:

  ```
  apiVersion: portal.apiconnect.ibm.com/v1beta1
  kind: PortalRestore
  metadata:
    generateName: portal-restore-
  spec:
    type: site
    portalCluster: <portal cluster name>
    siteName: <portal site to restore>
  ```

  where:

  - `portalCluster` is the cluster name that you identified in step 3.
  - `siteName` is either:
    - The UUID or URL of the site you want to restore, for example: `myportal.com`, in this case the latest backup of this site restored.
    - The backup ID of the backup that contains the site that you want to restore.
    - `all` - restore all sites in the latest backup.
    - `installed` - restore from the latest backup only sites that currently exist. For cases where your backup contains site backups for sites that no longer exist on your portal.
    - Backup .tgz file name. For example: site-1234.tgz restores the site from the backup file name. The file name of the backup is shown in the last column of the `kubectl get portalbackup -o wide` output.

- Restore type `system`. Restore portal system backup, but no sites.

  ```
  apiVersion: portal.apiconnect.ibm.com/v1beta1
  kind: PortalRestore
  metadata:
    generateName: portal-restore-
  spec:
    type: system
    portalCluster: <portal cluster name>
    systemName: <backup ID>
  ```

  where:

  - `portalCluster` is the cluster name that you identified in step 3.
  - `systemName` is either:
    - The backup ID of a portal backup CR. You can restore only portal database backups that are of `type` system and `crType` record.
    - Filename of the backup's .tgz file. Restores the system from the specified backup file name. The file name of the backup is shown in the last column of the `kubectl get portalbackup -o wide` output.
    - `latest`: restores from latest system backup that is available.

5. Optional: If the `API` hostname of your management subsystem platform API changed since taking the backup you are restoring, then add the property `customPlatformApiHostname` to your portalrestore_cr.yaml file and set it to the new platform API hostname of the management subsystem. For example:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalRestore
   metadata:
     generateName: portal-restore-
   spec:
     ...
     customPlatformApiHostname: mgmt.platformapi.host
     ...
   ```

6. Create the `PortalRestore` CR from the portalrestore_cr.yaml file:

   ```
   kubectl -n <portal namespace> create -f portalrestore_cr.yaml
   ```

7. Verify that the restore is running with:

   ```
   kubectl -n <portal namespace> get portalrestore
   ```

   Example output for a restore type of `all`:

   ```
   NAME                   TYPE    TASK ID            STATUS    MESSAGE                                                  AGE
   portal-restore-2xsgl   all     t3wz85rxz61jam92   Running   Sites Restored: 2, Sites Failed: 0, Sites Pending: 1
   ```

   Example output for a restore type of `system`:

   ```
   NAME                   TYPE     TASK ID             STATUS   MESSAGE                                                 CLUSTER
   AGE
   portal-restore-xtjjt   system   yrwtcw5ult08p786    Ready    Please describe the CR to see the full message.   portal
   72s
   ```

8. To see more information about the restore, run:

   ```
   kubectl -n <portal namespace> get portalrestore <restore name> -o yaml
   ```

Example output:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-11-11T10:14:39Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-8f4c5
  namespace: portal
  resourceVersion: "1497597"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-8f4c5
  uid: 5a70b657-d998-4989-b9b6-8480e986311d
spec:
  siteName: "20200526.090209"
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestorePending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    message: |-
      2020-05-26 13:11:25: CLI task (restore:site) starting.
      2020-05-26 13:11:36: Deleting any old site data if it exists...
      2020-05-26 13:13:18: Restoring site test.com ...
      2020-05-26 13:13:18: .
      2020-05-26 13:13:19: Patching the database dump before restoring ...
      2020-05-26 13:13:48: .
      2020-05-26 13:19:41: Triggering the drush command to restore the site. This may take some time.
      2020-05-26 13:19:48: .
      2020-05-26 13:25:40: Drush restore command completed. Setting up the site.
      2020-05-26 13:25:48: .
      2020-05-26 13:30:49: Site test.com restored from backup /var/aegir/backups/test.com-20200526.090209.tar.gz.
Initializing the site...
      2020-05-26 13:30:52: Restore site completed.
      2020-05-26 13:30:52: CLI task (restore:site) completed successfully.
    reason: RestoreComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    reason: RestoreInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: bybyibd99wxtmkmk
  message: ""
```

9. When the restore completes `RestorePortal` CR shows Ready state, regardless of whether the restore was successful or not.

```
NAME                    TYPE    TASK ID             STATUS    MESSAGE                                                    AGE
portal-restore-2xsgl    all     t3wz85rxz61jam92    Ready    Sites Restored: 3, Sites Failed: 0, Sites Pending: 0
```

10. If a site fails to restore, check for messages in the restore description:

    ```
    kubectl -n <portal namespace> get portalrestore <restore name> -o yaml
    ```

    Also, check the logs of the admin container in the **www** pod:

    ```
    kubectl -n <portal namespace> logs <portal www pod> -c admin
    ```

Recovering from a disaster

Reinstall the portal subsystem by using your infrastructure configuration backup, then restore the portal database backup.

Before you begin

To successfully recover the portal subsystem, you must have the following backups:

- Infrastructure configuration backup.
- Portal database backup.

About this task

- Portal subsystem backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore a portal subsystem backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see Migrating from v10 to v10 on a different form factor.
  Note: If your management platform API endpoint changed since the portal database backup was created, you can update this setting during portal database restore by using the `customPlatformApiHostname` property in the `PortalRestore` CR. This is the only exception to the identical endpoints requirement. See Restore portal database.
- When you restore a portal database backup, you must first restore the database backup of the management subsystem that the portal is registered with. Portal and management databases must remain in sync.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Reinstall your portal subsystem by using the YAML files from your Infrastructure configuration backup.
   Your infrastructure configuration backup consists of:
   - Portal CR YAML: portal_cr_backup.yaml.
   - Portal database encryption secret: portal_enc_key_backup.yaml.
   - Portal backup server credentials secret: ptl-backup-secret_backup.yaml.
   - If required, the portal backup server CA certificate secret: portal_custom_s3_ca.yaml.
   - If you customized any portal certificates, a YAML file of the secret for each customized certificate. For example: portalWeb_secret_backup.yaml.

   a. Create the database encryption secret with the portal_enc_key_backup.yaml:

      ```
      kubectl -n <portal namespace> apply -f portal_enc_key_backup.yaml
      ```

   b. Create the backup credential secrets with your ptl-backup-secret_backup.yaml file:

      ```
      kubectl -n <portal namespace> apply -f ptl-backup-secret_backup.yaml
      ```

   c. Create the S3 CA secret (if required) with:

      ```
      kubectl -n <portal namespace> apply -f portal_custom_s3_ca.yaml
      ```

   d. Create the secrets for any customized certificates from their backup files:

      ```
      kubectl -n <portal namespace> apply -f <custom certificate secret backup file>
      ```

   e. Install the portal subsystem. Follow the installation procedures for your platform: Deployment procedures, but use the settings from the `spec` section of your portal_cr_backup.yaml file.
      Note:
      - Cloud Pak for Integration users: Many of the portal configuration settings can be configured in Cloud Pak for Integration Platform UI. You do not need to edit the YAML directly.
2. Restore your portal database: Restoring from a portal database backup.

Troubleshooting portal database backup and restore

Troubleshoot problems with portal database backups and restores.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.
Check the status section of the PortalBackup or PortalRestore CR:

```
kubectl -n <portal namespace> get <portalbackup/portalrestore> <bup-name/restore-name> -o yaml
```

Example output:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-05-26T13:10:46Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-x2bf9
  namespace: portal
  resourceVersion: "1493471"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-x2bf9
  uid: 9f6a1e22-55ab-4595-8b98-bdfaec40d58c
spec:
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-05-26T13:10:49Z"
    message: |
      Please provide a site to restore. To restore all sites use value 'all' in the CR
    status: Failed
    type: Ready
  fullBackupName: ""
  id: ""
  message: ""
```

Check operator pod logs. Look for information in the logs of the `ibm-apiconnect` pod:

- Locate the `ibm-apiconnect` pod with:

  ```
  kubectl get pods -n <namespace-of-portal-subsystem>
  ```

- View the logs with:

  ```
  kubectl logs <ibm-apiconnect-pod-NAME> -n <namespace-of-portal-subsystem>
  ```

Common problems with backup and restore

These are some frequent issues that can cause backup or restore to fail:

- Invalid login credentials for your remote backup server. Check that the username and password that you configured in your subsystem database backup settings is correct.
- The user does not have write permissions on the backup server. Check that the username you configured in your subsystem database backup settings has write permission to the specified backup path.
- Remote backup server storage full. Check your remote backup server and if the storage is full either extend the storage space or delete older backups.
- No network access to the remote backup server. Check that you can communicate with your remote backup server from your API Connect environment.
- TLS handshaking failure with the remote backup server. If your remote backup server has a self-signed CA certificate, check that this certificate is trusted by your API Connect deployment, see the database backup configuration steps for your subsystem for more information.

Backing up and restoring the analytics subsystem

How to configure analytics backups and recover from a disaster.

For information about analytics backup, restore, and disaster recover, refer to Configuring analytics backups.

Backing up and restoring on OpenShift top-level CR

How to backup and restore your top-level CR deployment of API Connect

- **Backing up your OpenShift top-level CR API Connect deployment**
- **Disaster recovery on OpenShift top-level CR**

Backing up your OpenShift top-level CR API Connect deployment

About this task

A full API Connect backup consists of:

- Infrastructure configuration backup: The infrastructure configuration backup includes everything that you configure for API Connect in your deployment environment, such as:
  - Custom resource (CR) YAML files.
  - Custom certificates.
  - Encryption secrets.
  - Backup secrets.
- Subsystem database backups: The management, portal, and analytics subsystems all have their own databases that must be backed up:
  - Management database contains everything users configure in the management UIs, REST API, toolkit CLI, and portal sites (excluding Drupal customizations).
  - Portal database contains all portal and site customizations.
  - Analytics database contains all API event data, and the settings of analytics features configured in the management UIs.

Procedure

1. Take a backup of your top-level CR:

   ```
   oc -n <namespace> get ApiconnectCluster -o yaml > apic_cr_backup.yaml
   ```

2. Take backups of each subsystem, following the documented steps:
   - Backing up and restoring the management subsystem
   - Backing up and restoring the developer portal
   - Backing up and restoring the analytics subsystem

Disaster recovery on OpenShift top-level CR

About this task

To recover your API Connect deployment, follow the steps in this topic along with the instructions for Installing with the top-level CR on OpenShift.

Note: The steps that are detailed in this topic use the Kubernetes `kubectl` command, which does work on OpenShift, but if you want you can replace `kubectl` with the OpenShift equivalent command: `oc`.

Procedure

1. Create the management subsystem Kubernetes secrets from the backup YAML files that you created in Management infrastructure configuration backup.
   a. Create the database encryption secret with the management_enc_key_backup.yaml file:

      ```
      kubectl -n <management namespace> apply -f management_enc_key_backup.yaml
      ```

   b. Create the backup credential secret with your mgmt-backup-secret_backup.yaml file:

      ```
      kubectl -n <management namespace> apply -f mgmt-backup-secret_backup.yaml
      ```

   c. Create the credential secrets with the <credential secret>_backup.yaml files you saved in step 5 of Taking a management infrastructure configuration backup:

      ```
      kubectl -n <management namespace> apply -f <credential secret backup file>
      ```

   d. Set the S3 CA secret (if required) from your backup file:

      ```
      kubectl -n <management namespace> apply -f mgmt_custom_s3_ca.yaml
      ```

   e. Create the secrets for any customized certificates from their backup files:

      ```
      kubectl -n <management namespace> apply -f <custom certificate secret backup file>
      ```

2. Create the portal subsystem Kubernetes secrets from the backup YAML files that you created in Portal infrastructure configuration backup.
   a. Create the database encryption secret with the portal_enc_key_backup.yaml:

      ```
      kubectl -n <portal namespace> apply -f portal_enc_key_backup.yaml
      ```

   b. Create the backup credential secrets with your ptl-backup-secret_backup.yaml file:

      ```
      kubectl -n <portal namespace> apply -f ptl-backup-secret_backup.yaml
      ```

   c. Create the S3 CA secret (if required) with:

      ```
      kubectl -n <portal namespace> apply -f portal_custom_s3_ca.yaml
      ```

   d.
   e. Create the secrets for any customized certificates from their backup files:

      ```
      kubectl -n <portal namespace> apply -f <custom certificate secret backup file>
      ```

3. Re-create the analytics secrets that you backed up. See Backup analytics secrets.
4. Prepare the top-level CR for your replacement API Connect, copy and paste the `spec` section from your apic_cr_backup.yaml file.
5. Create the API Connect deployment from the YAML file.
   Refer to the installation steps in Installing with the top-level CR on OpenShift.
   Wait for all subsystems to be created and ready. Monitor the installation progress with the command:

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

6. Restore the subsystem databases:
   a. Restoring the management database with a backup ID.
      Verify that the restoration completed successfully and that the management subsystem is Ready, then proceed to the next step and restore the portal subsystem.

   b. Restoring from a portal database backup.
      Use the restore type `all` to ensure that you restore the complete subsystem and all portal sites.

   c. Restoring the analytics database.

Backing up and restoring on Cloud Pak for Integration and OpenShift top-level CR

How to backup and restore all API Connect subsystem in a Cloud Pak for Integration, or OpenShift top-level CR deployment

- **Backing up your Cloud Pak for Integration API Connect deployment**
  How to backup your API Connect configuration and subsystem databases.
- **Recovering from a disaster on Cloud Pak for Integration**
  Prepare your Cloud Pak for Integration environment for installation of the replacement API Connect deployment. Install API Connect. Restore the database backups of each subsystem.

Backing up your Cloud Pak for Integration API Connect deployment

How to backup your API Connect configuration and subsystem databases.

About this task

A full API Connect backup consists of:

- Infrastructure configuration backup: The infrastructure configuration backup includes everything that you configure for API Connect in your deployment environment, such as:
  - Custom resource (CR) YAML files.
  - Custom certificates.
  - Encryption secrets.

    - Backup secrets.
- Subsystem database backups: The management, portal, and analytics subsystems all have their own databases that must be backed up:
    - Management database contains everything users configure in the management UIs, REST API, toolkit CLI, and portal sites (excluding Drupal customizations).
    - Portal database contains all portal and site customizations.
    - Analytics database contains all API event data, and the settings of analytics features configured in the management UIs.

To backup your Cloud Pak for Integration or OpenShift top-level CR deployment, backup the infrastructure configuration and databases of each subsystem:

Procedure

1. Save the CP4I credentials secret.

   ```
   oc -n <namespace> get secret <instance_name>-cp4i-creds -o yaml > cp4i_credentials_secret_backup.yaml
   ```

   Edit the saved YAML file and remove both the **ownerReferences** subsection and the **selfLink** property, and then save the updated file.

2. If you did not change the cloud administrator password after installation, backup the cloud admin password secret.

   ```
   oc -n <namespace> get secret <instanceName>-mgmt-admin-pass -o jsonpath="{.data.password}" | base64 -d
   ```

3. Backup the API Connect CR.

   ```
   oc -n <namespace> get ApiconnectCluster -o yaml > apic_cr_backup.yaml
   ```

4. Follow the backup steps for each subsystem.
    - Backing up and restoring the management subsystem
    - Backing up and restoring the developer portal
    - Backing up and restoring the analytics subsystem

   Important: Ensure that you backup all subsystem CRs, and not just the top-level CR. The subsystem CRs contain additional information about each subsystem, which is required for disaster recovery.

Recovering from a disaster on Cloud Pak for Integration

Prepare your Cloud Pak for Integration environment for installation of the replacement API Connect deployment. Install API Connect. Restore the database backups of each subsystem.

About this task

To recover from a disaster, re-create your API Connect deployment, referring to the backup files and artifacts you took when you followed the backup process: Backing up your Cloud Pak for Integration API Connect deployment.

Note: Your recovery installation of Cloud Pak for Integration must include the Platform UI if your original deployment used it. In addition, the CR must use the same settings as in the original deployment.

Procedure

Preparing the Cloud Pak for Integration environment.

1. Prepare your Cloud Pak for Integration environment for deploying API Connect as explained in the appropriate version of the Cloud Pak for Integration documentation.
   After you prepare your environment, return to this procedure to configure the installation CR, deploy API Connect, and then restore the subsystem database backups to your API Connect subsystems.

2. Create management and portal encryption secrets from your backups of these secrets.
    a. Create the management database encryption secret.

       ```
       oc -n <namespace> apply -f management_enc_key_backup.yaml
       ```

       where management_enc_key_backup.yaml is the encryption secret backup that you created in step 2.c of Taking a management infrastructure configuration backup
    b. Create the portal database encryption secret.

       ```
       oc -n <namespace> apply -f portal_enc_key_backup.yaml
       ```

       where portal_enc_key_backup.yaml is the encryption secret backup that you created in step 2.c of Taking a portal infrastructure configuration backup.
3. Create the Cloud Pak for Integration credentials secret.

   ```
   oc -n <namespace> apply -f cp4i_credentials_secret_backup.yaml
   ```

   where cp4i_credentials_secret_backup.yaml is the credentials backup file that you created in step 1 of Backing up your Cloud Pak for Integration API Connect deployment.
4. Generate the Kubernetes secret for each subsystem's database backups, by using the backup files that you created for each subsystem.
   For each secret, use the same name that you used in the original deployment (you can see the name in the **credentials** setting of each subsystem's **databaseBackup** section).

    a. `oc -n <namespace> apply -f mgmt-backup-secret_backup.yaml`

       where mgmt-backup-secret_backup.yaml was created in step 3 on Taking a management infrastructure configuration backup.

    b. `oc -n <namespace> apply -f ptl-backup-secret_backup.yaml`

where ptl-backup-secret_backup.yaml was created in step 3 on Taking a portal infrastructure configuration backup.

c. `oc -n <namespace> apply -f analytics-backup-secret_backup.yaml`

where analytics-backup-secret_backup.yaml was created in Configuring analytics backups.

5. Create each of the management client application credential secrets.
Run the following command to apply each secret, by using the backup YAML files that you created in step 5 of Taking a management infrastructure configuration backup.

`oc -n <namespace> apply -f <secretName>.yaml`

Example:

```
atmCredentialSecret: management-atm-cred
consumerToolkitCredentialSecret: management-ccli-cred
consumerUICredentialSecret: management-cui-cred
designerCredentialSecret: management-dsgr-cred
discoveryCredentialSecret: management-discovery-cred
governanceCredentialSecret: management-governance-cred
juhuCredentialSecret: management-juhu-cred
toolkitCredentialSecret: management-cli-cred
uiCredentialSecret: management-ui-cred
```

Prepare draft API Connect installation YAML file.

6. Log in to the IBM Cloud Pak Platform UI and create a draft API Connect YAML.
In this step, you complete the initial steps for deploying API Connect in Cloud Pak for Integration, but you do not begin the actual installation until step 8. Instead, you update the draft API Connect YAML to match the API Connect YAML you backed up in Backing up your Cloud Pak for Integration API Connect deployment.

a. Log in to the Platform UI.
b. On the home page, click Create an instance.
c. Select the API management tile and click Next.
d. On the Create an instance of API management page, select the deployment type, and click Next.
Choose the same deployment type that you used when you originally deployed API Connect.

e. On the deployment settings page, click the YAML tab to edit the installation CR in YAML format.

Update draft API Connect YAML with values from your CR backup files.

7. Refer to your CR backup files apic_cr_backup.yaml and *<subsystem>*_cr_backup.yaml to update the properties in your draft YAML.
a. Set `metadata.name` to the value from your backup apic_cr_backup.yaml.
b. If your original `metadata.name` was 10 characters or longer, you must set the `spec.<subsystem>.originalUID` to the values from your management and portal CR backups.
The IDs were created with the original deployment of the management and portal subsystems, and the replacement subsystems must use the same IDs. The IDs are in the `spec.originalUID` of the mgmt_cr_backup.yaml and portal_cr_backup.yaml backup files.

For example:

```
spec:
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
  portal:
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

c. Set the management and portal encryption secrets.
Set `spec.<subsystem>.encryptionSecret.secretName` with the secrets you created in step 2, as shown in the following example:

```
spec:
  management:
    encryptionSecret:
      secretName: mgmt-enc-key
  portal:
    encryptionSecret:
      secretName: portal-enc-key
```

d. Set each of the credential secrets in `spec.management` section, to the secrets you created in step 5.
Example:

```
spec:
  management:
    customApplicationCredentials:
    - name: atm-cred
      secretName: management-atm-cred
    - name: ccli-cred
      secretName: management-ccli-cred
    - name: cui-cred
      secretName: management-cui-cred
    - name: dsgr-cred
      secretName: management-dsgr-cred
    - name: governance-cred
      secretName: management-governance-cred
    - name: juhu-cred
      secretName: management-juhu-cred
    - name: cli-cred
      secretName: management-cli-cred
    - name: ui-cred
      secretName: management-ui-cred
```

e. Set the `siteName` property in the `spec.management` and `spec.portal` sections, by copying the values from your mgmt_cr_backup.yaml and portal_cr_backup.yaml backup files.

For example:

```
spec:
  management:
    siteName: "a2a5e6e2"
  portal:
    siteName: "890772e3"
```

f. If you are restoring your deployment in a different data center, explicitly set your endpoints to prevent auto-generation of different endpoints.
If you are restoring your deployment in a different data center, the endpoints that are used in your original deployment must be the same in your new data center. The Platform UI in Cloud Pak for Integration auto-generates the endpoints if they are left empty in the YAML, therefore ensure that you explicitly set the endpoints in the CR to match what is set in your <subsystem>_cr_backup.yaml files.

For example:

```
apiManagerEndpoint:
      annotations:
        cert-manager.io/issuer: prod-ingress-issuer
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: prod-api-manager.example.com
        secretName: prod-be787dd3-api-manager
```

g. Set the subsystem database backup and restore settings to match what you have in apic_cr_backup.yaml.
The **spec.*<subsystem name>*.databaseBackup** sections for each subsystem contain the database backup settings.

Start API Connect deployment.

8. On the deployment page, install API Connect by clicking Create.
Wait for all subsystems to be created and ready. Monitor the installation progress with the command:

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

9. Restore the subsystems' database backups in the following sequence:
   a. Restoring the management database with a backup ID.
   Verify that the restoration completed successfully and that the management subsystem is Ready, then proceed to the next step and restore the portal subsystem.

   b. Restoring from a portal database backup.
   Use the restore type **all** to ensure that you restore the complete subsystem and all portal sites.

   c. Restoring the analytics database.
10. Force the configurator to run again.
The cluster will still not be ready after restore at this stage because the configurator has yet to successfully complete. For the configurator to run again, delete the associated job so that a new pod starts running:

    a. Run the following command to get the list of jobs:

    ```
    oc get jobs -n <APIC_namespace>
    ```

    b. Run the following command to determine the name of your API Connect instance:

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

    c. Run the following command to delete the configurator job:

    ```
    oc -n <APIC_namespace> delete job <instance_name>-configurator
    ```

11. Verify that the recovery was successful:
    a. Ensure that you can log in to the Cloud Manager UI.
    b. Verify that your provider organizations exist.
    c. Ensure that you can log in to each portal site.
    d. Ensure that the analytics dashboard contains your restored analytics data.

Backing up, restoring, and disaster recovery V10.0.6.0

You can backup and restore API Connect subsystems.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. API Connect Version 10 uses Kubernetes operators to back up and restore databases.

- Backups are for recovery of the API Connect subsystems in the same environment that the backups were taken. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see Migrating from v10 to v10 on a different form factor.
- Manual backups
For the management database, the backup configuration is specified in a Management CR (custom resource). You create the backup CR manually. Once the backup CR is created, the **apiconnect** operator performs backup and updates the status into the backup CR.

- Scheduled backups
For scheduled backups, you can add a backup schedule to the backup configuration details in the management CR. The apiconnect operator will then trigger and perform backups as specified in the custom **cron** schedule. A backup CR is created for every individual backup invocation as described in the cron schedule. You can view each backup CR to determine the status of each backup.

- Restore
For restore, you update the backup Management CR. You then create a restore CR with the backup name in it. You can list the backup CRs and determine the backup name by looking at individual backup CR. The **apiconnect** operator performs the restore.

- You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.
- If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.
- When restoring, the management, analytics, and developer portal subsystems must be at the same version and fix pack level.

Note: For information about how to maintain a two data center deployment, see Maintaining a two data center deployment.

- **Backups on Kubernetes V10.0.6.0**
  Use the procedures in this section to backup and restore API Connect on generic Kubernetes distributions.
- **Backups on OpenShift V10.0.6.0**
  You can backup and restore API Connect subsystems on OpenShift and Cloud Pak for Integration.
- **Disaster recovery V10.0.6.0**
  Prepare for, or recover from, a disaster affecting an API Connect deployment running on Kubernetes, OpenShift, or Cloud Pak for Integration.

Backups on Kubernetes V10.0.6.0

Use the procedures in this section to backup and restore API Connect on generic Kubernetes distributions.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- **Backing up and restoring the management subsystem v10.0.6.0**
  You can use the following instructions to back up and restore your Management subsystem on API Connect.
- **Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0**
  You can back up and restore your Developer Portal service in your Kubernetes environment.
- **Backing up and restoring the analytics database**
  The API Connect analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the management subsystem v10.0.6.0

You can use the following instructions to back up and restore your Management subsystem on API Connect.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- To complete a backup and restore of your Management subsystem, you must have the `custom resource` configured with the `databaseBackup` subsection in the management subsystem custom resource.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required:

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

- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.
- Custom resource (CR) templates are provided in the `helper_files` archive to install each API Connect subsystem. You decompressed these files into an installation folder of your choosing in Obtaining product files.
- For points to consider when backing up and restoring a two data center disaster recovery deployment on Kubernetes, see Backup and restore requirements for a 2DCDR deployment.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.

Note: You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported

- Cloud-based S3 storage
  - IBM Cloud Object Storage
  - AWS S3
  - **v10.0.2.0 or greater:** Any custom S3, such as minIO.
- SFTP
- Local storage backups

- **Configuring backup settings for fresh install of the Management subsystem v10.0.6.0**
  You can configure backups for your Management subsystem in your Kubernetes environment.
- **Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0**
  You can reconfigure the backup settings after installation of the management subsystem.
- **Generating a manual backup of the management subsystem v10.0.6.0**
  You can generate a manual (non-scheduled) backup of the management subsystem
- **Restoring the management subsystem v10.0.6.0**
  You can restore your Management subsystem in your Kubernetes environment.
- **Troubleshooting management subsystem backups v10.0.6.0**
  You can troubleshoot failures of the management subsystem backups.

Configuring backup settings for fresh install of the Management subsystem v10.0.6.0

You can configure backups for your Management subsystem in your Kubernetes environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Use the instructions for your backup type:

- **Configuring S3 backup settings for fresh install of the Management subsystem v10.0.6.0**
  You can configure backups for your management subsystem in your Kubernetes environment.
- **Configuring SFTP backup settings for fresh install of the Management subsystem v10.0.6.0**
  You can configure backups for your Management subsystem in your Kubernetes environment.
- **Configuring local backup settings for fresh install of the Management subsystem v10.0.6.0**
  You can configure backups for your Management subsystem in your Kubernetes environment.

Configuring S3 backup settings for fresh install of the Management subsystem v10.0.6.0

You can configure backups for your management subsystem in your Kubernetes environment.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Review Backing up and restoring the management subsystem v10.0.6.0.
If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:

1. Remove the **multiSiteHA** section from the **spec** section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the **multiSiteHA** sections that you remove.
2. Configure your S3 backup settings. You must specify different backup locations for each data center, see Backup and restore requirements for a 2DCDR deployment.
3. Add the **multiSiteHA** sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

About this task

If you haven't already, configure your management subsystem **custom resource** with the **databaseBackup** subsection.

Note: **S3Provider** supports **custom** S3 solutions in 10.0.2.0 and greater. The support includes new backup parameters **backupCerts** and **backups3URIStyle**.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions that can cause database restores to fail.
Warning:
When configuring a new management subsystem for S3 backup, the content of the S3 bucket path must be empty. If you want to configure a management subsystem to use the content of an S3 bucket path already used by another management subsystem, you have two options:

1. Shutdown the postgres database on the existing management subsystem that points to the S3 bucket path before you start the new management subsystem.
2. In a disaster recovery scenario where you want to restore from the backup of an existing installation: Create a new S3 path and use S3 tools to copy the backup content from the existing management subsystem S3 bucket path, to the new bucket path.

Procedure

1. Create a backup secret.
   The backup secret is a Kubernetes secret that contains your username and password for your S3 backup database.

   For examples of how to generate the appropriate access key and secret for two of them, see:

   - IBM (Cloud Object Storage): Service credentials.
   - AWS: Managing access keys.
   - Custom: See documentation for your S3 provider. For example, minIO.

   The secret can be created with the following command:

   ```
   $ kubectl create secret generic mgmt-backup-secret --from-literal=key='<YOUR ACCESS KEY>' \
     --from-literal=keysecret='<YOUR ACCESS KEY SECRET>' -n <namespace_of_management_subsystem>
   ```

2. Ensure that your Management subsystem **custom resource** is configured with the **databaseBackup** subsection.
   For example:

   ```
   databaseBackup:
     protocol: objstore
     s3provider: ibm
     host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
     path: apic-backup
     credentials: mgmt-backup-secret
     backupCerts: <custom-s3-server-CA-cert>
     backups3URIStyle: host(default)/path
     schedule: "0 3 * * *"
     repoRetentionFull: 2
   ```

   Note: backupCerts and backups3URIStyle are used for custom s3 only, on **v10.0.2.0 or greater**.

Table 1. Backup configuration settings

Setting	Description

Setting	Description
`protocol`	The type of the backup. For s3 storage: `objstore`. When the parameter is not set, the backup type defaults to local storage backup.
`s3provider`	Name of the S3 provider to use. You must specify one of the supported values of `aws`, `ibm`, or `custom`. Support for `custom` is available in **v10.0.2.0 or greater**. Note: The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: `x509: certificate signed by unknown authority`.
`host`	For `objstore` type backup, specify S3 endpoint with the corresponding S3 region in the format `<S3endpoint>/<S3region>`
`path`	The path to the location of the backup: For `objstore` backups, the name of your S3 bucket to store backup data. For example, `bucket-name/folder`. The use of subdirectories in the bucket name is not supported. Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name. Ensure that `bucket-name/folder` is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. **Explanation:** Once a folder is used, archive.info and backup.info files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring.
`credentials`	For `objstore` type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret

`backupCerts`	**v10.0.2.0 or greater**
	
	Custom certificate. Used only when `s3Provider` is set to `custom`. Most of the custom S3 providers are created based on custom CA certificate.
	
	This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be `ca.crt` and value should be base64 encoded value of CA certificate.
	
	The API Connect management subsystem validates the custom S3 certificate against the customer-provided CA certificate.
	
	Sample bash script to create the Kubernetes secret:
	
	`cat >customs3ca.yaml <<EOF`
	`apiVersion: v1`
	`data:`
	`  ca.crt: $(base64 <path-to-ca-certificate> | tr -d '\n' )`
	`kind: Secret`
	`metadata:`
	`  name: custom-server-ca`
	`type: generic`
	`EOF`
	
	`kubectl apply -f customs3ca.yaml -n <namespace>`

`backups3URIStyle`	**v10.0.2.0 or greater** Valid values: `host` and `path`. Only allowed if `s3Provider` is set to `custom`. Some custom S3 providers require URI style to be set to `path`. For example, minIO supports both `host` and `path` style setup. You can create a minIO S3 server to only accept `path` style client communications. **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, upstream custom S3 can reject connections from the client, such as API Connect management subsystem.
`schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. There is no default backup schedule set. Be sure to set your backup schedule. All scheduled Management subsystem backups are of type `full` only.
`repoRetentionFull`	**v10.0.3.0 or later** - The number of full S3 backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: `1` Maximum value: `9999999` Default: none

3. Verify that the configuration deploys successfully.
   When you configure Management subsystem backups, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

   Check the status of the `stanza-create` job:

   `kubectl get jobs -n <namespace> | grep stanza`

- On success:
  - The `stanza-create` job status is `1/1`:

    ```
    kubectl get jobs -n <namespace> | grep stanza
    m1-b722e361-postgres-stanza-create          1/1           5s         127m
    ```

  - The `stanza-create` job shows `status: "True"` and `type: Complete`:

    ```
    kubectl get job -n <namespace> m1-b722e361-postgres-stanza-create -o yaml

    status:
      completionTime: "2021-04-13T18:38:50Z"
      conditions:
        - lastProbeTime: "2021-04-13T18:38:50Z"
          lastTransitionTime: "2021-04-13T18:38:50Z"
          status: "True"
          type: Complete
     startTime: "2021-04-13T18:38:45Z"
     succeeded: 1
    ```

  - The `stanza-create` pod is in `Completed` state:

    ```
    kubectl get pods -n <namespace> | grep stanza
    m1-b722e361-postgres-stanza-create-q4fvp                      0/1     Completed   0          127m
    ```

  - Exec into the pod:

    ```
    kubectl -n <namespace> exec -it <backrest-shared-repo-pod> -- bash
    ```

  - The `pgbackrest` command returns the S3 contents in valid JSON:

    ```
    pgbackrest info --output json --repo1-type s3

    "backup":{"held":false}},"message":"ok"}][pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdfdd5-rs882
    /
    ```

- On failure, see Troubleshooting stanza-create job for S3 backup configuration v10.0.6.0.

Configuring SFTP backup settings for fresh install of the Management subsystem v10.0.6.0

You can configure backups for your Management subsystem in your Kubernetes environment.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Review Backing up and restoring the management subsystem v10.0.6.0.

About this task

If you haven't already, configure your Management subsystem `custom resource` with the `databaseBackup` subsection.

Procedure

1. Create a backup secret.
   The backup secret is a Kubernetes secret that contains your credentials for accessing the SFTP backup database. Supported credentials types:

   - Username and password (v10.0.2.0 or later).
   - Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.[1]

   Use one of the following commands to create the secret:

   - Username and password credentials

     ```
     $ kubectl create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>'
     --from-literal=password='<YOUR PASSWORD>' -n <namespace-of-mgmt-subsystem>
     ```

   - **Version 10.0.3.0 or later**: Username and SSH-key credentials:

     ```
     $ kubectl create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>'
      --from-file=ssh-privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-mgmt-subsystem>
     ```

   Note: If the username contains the backslash character, insert an extra backslash to escape it. For example, if the username is:

   ```
   backup\svc-apic
   ```

   set this as:

   ```
   -from-literal=username='backup\\svc-apic'
   ```

2. Ensure that your Management subsystem `custom resource` is configured with the `databaseBackup` subsection.
   For example:

```
databaseBackup:
  protocol: sftp
  host: <SFTP-host-name>
  port: <SFTP-port>
  path: apic-backup
  retries: 0
  credentials: mgmt-backup-secret
  schedule: "0 3 * * *"
```

Table 1. Backup configuration settings

Setting	Description
`protocol`	The type of the backup. For SFTP storage: **`sftp`**.
`host`	The backups host. For **`sftp`** type, the SFTP server hostname
`port`	The SFTP server port. Optional. Default: **`22`**.
`path`	The path to the location of the backup. For **`sftp`** type, the full absolute path of the folder on the SFTP server, beginning with /.
`retries`	The number of times the **`ibm-apiconnect`** Operator attempts backups in the event of a failed SFTP backup. Default value: **`0`**.
`credentials`	Name of the Kubernetes secret containing your SFTP Username/password. For Version 10.0.3.0 or later, the Kubernetes secret can contain your SFTP Username and SSH key.
`schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------ min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. There is no default backup schedule set. Be sure to set your backup schedule. All scheduled Management subsystem backups are of type **`full`** only.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Configuring local backup settings for fresh install of the Management subsystem v10.0.6.0

You can configure backups for your Management subsystem in your Kubernetes environment.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Review Backing up and restoring the management subsystem v10.0.6.0.

About this task

If you haven't already, configure your Management subsystem **`custom resource`**.

Procedure

Configure your Management subsystem **`custom resource`** with the **`databaseBackup`** subsection.
Specify **`schedule`** and **`repoRetentionFull`**, the default settings are 01:00 UTC and 14 days.

```
databaseBackup:
  schedule: "0 1 * * *"
  repoRetentionFull: 14
```

Table 1. Backup configuration settings

Setting	Description

Setting	Description
`schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------ min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. The default is daily backups at 01:00 UTC. All scheduled Management subsystem backups are of type `full` only.
`repoRetentionFull`	- The number of full local backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: `1` Maximum value: `9999999` Default: 14

Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0

You can reconfigure the backup settings after installation of the management subsystem.

Procedure

1. Make sure that the management cluster status is `Running` and the READY condition displays the same value before and after the "/".
   For example:

   ```
   k get mgmt
   NAME   READY   STATUS    VERSION      RECONCILED VERSION   AGE
   mgmt   16/16   Running   10.0.6.0   10.0.6.0-xxxx       19h
   ```

2. Create backup secret as described in Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
3. Complete the following steps, based on your storage type:
   - S3 storage
     You can change or add the `DatabaseBackup` s3 configuration via the Management Cluster CR after fresh install. However, doing so requires a reconfiguration of the database and a brief downtime. The operator restarts the database with the new backup configurations as soon as the new configuration in the management CR is saved.

     If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:
     - Remove the `multiSiteHA` section from the `spec` section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the `multiSiteHA` sections that you remove.
     - Configure your S3 backup settings. You must specify different backup locations for each data center, see Backup and restore requirements for a 2DCDR deployment.
     - Add the `multiSiteHA` sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

     Note:
     If you have an APIConnectCluster custom resource installed, do not modify the ManagementCluster CR. Instead, modify the APIConnectCluster CR.

     You can verify if you have an `apiconnectcluster` CR with the command

     ```
     $ kubectl get apiconnectcluster -n <your-namespace>
     ```

     Configuration of s3 backups uses a new parameter `restartDB` is required when changing the backup configuration. This parameter is in addition to the parameters described in Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.

     Example CR:

     ```
     databaseBackup:
       protocol: objstore
       s3provider: ibm
       host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
       path: apic-backup
       credentials: mgmt-backup-secret
       schedule: "0 3 * * *"
       restartDB:
         accept: true
     ```

Table 1. Additional parameter for S3 backup type reconfiguration

Parameter	Description
`restartDB:` `accept: <value>`	Required when changing the backup configuration. The parameter `accept` is a boolen: <`value`> can be `true` or `false`. When `accept` is `true`, and a new configuration for management CR is saved, the operator: • Stops the database and any dependent services • Applies the change • Restarts the stopped components. Note: `restartDB` is not required for initial deployments, only for modifications after the cluster is deployed.

- SFTP
  You can change/add the `DatabaseBackup` SFTP configuration via the Management Cluster CR after fresh install. This process does not involve any downtime. The same backup settings are used for reconfiguring as for initial configuration. For complete description of backup settings, see Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.

  Example CR:

```
protocol: sftp
host: <SFTP-hostname>
port: <SFTP-port>
path: apic-backup
retries: 0
credentials: mgmt-backup-secret
schedule: "0 3 * * *"
```

Generating a manual backup of the management subsystem v10.0.6.0

You can generate a manual (non-scheduled) backup of the management subsystem

Procedure

1. Make sure that the management cluster status is Running and the READY condition displays the same value before and after the "/".
   For example:

```
kubectl get mgmt
NAME    READY   STATUS    VERSION      RECONCILED VERSION   AGE
mgmt    16/16   Running   10.0.2.0   10.0.2.0-xxxx        19h
```

   Note: Be sure `READY` status is `16/16`. If you have just configured or reconfigured the backup settings, there is a period of time when the postgres pods are taken offline and brought back up with the new configuration. During this time, the `kubectl get mgmt` command will show `11/11` instead of `16/16`. If you start a backup before the Management cluster is on `16/16`, the backup attempt might stall indefinitely.
2. Decide which of the two types of manual backups you want to do:
   - `full`: backs up the entire Management Subsystem database.
   - `incr`: backs up any data that has not been backed up since the last `full` or `incr` backup.
3. You generate a backup manually by creating a managementBackup custom resource which is detected by the ibm-apiconnect operator that then triggers the backup process.
   You can use `mgmtbackup_cr.yaml` in the `helper_files` directory as an example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  generateName: management-
spec:
  type: full
  crType: create
  clusterName: management
```

   Table 1 lists the CR settings for a manual backup.

Table 1. CR settings for manual backup

Setting	Description
`type`	The type of backup to be done. Valid arguments are: • `full` Backs up the entire Management subsystem database • `incr` Backs up any data that has not been backed up since the last `full` or `incr` backup.
`crType`	Use `create` to trigger a backup process.
`clusterName`	The name of the target ManagementCluster you want to backup.

4. Create the ManagementBackup custom resource in the namespace of your Management Subsystem to trigger the backup process:

```
$ kubectl create -f mgmtbackup_cr.yaml -n <namespace-of-mgmt-subsystem>
```

5. (Optional): You can list current backups by using the command:

```
$ kubectl get managementbackup -n <namespace-of-mgmt-subsystem> --sort-by=.metadata.creationTimestamp
```

   Here is an example of some possible output:

```
NAME                STATUS   ID                 CLUSTER   TYPE   CR TYPE   AGE
m1-e46bbac4         Ready    20201023-183005F   m1        full   record    66m
mgmt-backup-dnvxw   Ready    20201023-193137F   m1        full   create    9m37s
```

6. (Optional): You can get a detailed view of a specific backup by using the command:

```
$ kubectl get managementbackup <bup-name> -n <namespace-of-mgmt-subsystem> -o yaml
```

Here is an example of some possible output:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  creationTimestamp: "2020-10-23T18:34:11Z"
  generation: 1
  labels:
    app.kubernetes.io/instance: m1
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: m1-e46bbac4
  name: m1-e46bbac4
  namespace: default
  resourceVersion: "6869"
  selfLink: /apis/management.apiconnect.ibm.com/v1beta1/namespaces/default/managementbackups/m1-e46bbac4
  uid: b7b6be9c-82d4-4737-984f-58584a3fe582
spec:
  clusterName: m1
  crType: create
  type: full
status:
  clusterName: m1-fbaa5be2-postgres
  conditions:
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupPending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    message: Record of management backup complete
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  crType: ""
  id: 20201023-183005F
  info:
    id: 20201023-183005F
    range: 2020-10-23 18:30:05 +0000 UTC - 2020-10-23 18:31:22 +0000 UTC
    rangeEnd: 1603477882
    rangeStart: 1603477805
    size: 32.6 MB
    totalsize: 32.6 MB
    type: full
  phase: Ready
  state: ""
  subsysName: m1
```

Restoring the management subsystem v10.0.6.0

You can restore your Management subsystem in your Kubernetes environment.

Before you begin

You must have backups of the management subsystem in order to restore. See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0 and Generating a manual backup of the management subsystem v10.0.6.0.

About this task

To trigger a restore, you will create a **ManagementRestore** custom resource. When the **ibm-apiconnect** operator detects the new CR, it starts the restore process.

- Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Management subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore, and that the backup ID in the selected backup exists in remote storage.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.
- If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.

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

Procedure

1. Obtain a list of the available backups, and decide which one to restore your Management subsystem:

```
kubectl get mgmtb -n  <your-namespace>
```

For example:

```
$ kubectl get mgmtb
NAME                  STATUS   ID                  CLUSTER      TYPE   CR TYPE   AGE
management-3aa3bebf   Ready    20200929-152150F    management   full   record    10h
management-3c4ca8df   Ready    20200929-115520F    management   full   record    20h
management-5tbxj      Ready    20200929-224349F    management   full   create    17h
management-f10af337   Ready    20200925-133703F    management   full   record    5d2h
management-jtlcd      Ready    20200929-224349F    management   full   create    25h
management-zxjtz      Ready    20200929-224349F    management   full   create    28h
```

Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.
You will use one of the values in the **NAME** column.

2. Configure `mgmtrestore_cr.yaml`. Use the copy in the `helper_files` directory as an example:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementRestore
metadata:
  generateName: management-
spec:
  backupName: management-3aa3bebf
```

`backupName`: is the name of the backup custom resource you want to restore to. Note that the sample uses example output from the previous step, to specify the backup named `management-3aa3bebf`:

```
backupName: management-3aa3bebf
```

3. Create the `ManagementRestore` CR in the namespace of the Management Subsystem to trigger the restore process:

```
$ kubectl create -f mgmtrestore_cr.yaml -n <namespace-of-mgmt-susbystem>
```

4. You can list current restores using the following command:

```
$ kubectl get managementrestore -n <namespace-of-mgmt-subsystem> --sort-by=.metadata.creationTimestamp
```

Here is an example of some possible output:

- SFTP:

```
NAME                  STATUS      BACKUP               CLUSTER      MESSAGE
PITR     AGE
mgmt-restore-ptx6d    Complete    mgmt-backup-w4h8c    m1           Restore process completed (DB Restore + DRR job)
6m31s
```

- S3

```
NAME                  STATUS      BACKUP               CLUSTER       MESSAGE
PITR                     AGE
mgmt-restore-hr9zx    Complete    mgmt-backup-xnjkl    m1            Restore process completed (DB Restore + DRR job)
2020-11-23 23:08:32+00   9m8s
```

Troubleshooting management subsystem backups v10.0.6.0

You can troubleshoot failures of the management subsystem backups.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- **Troubleshooting resiliency issues v10.0.6.0**
  Resolve resiliency issues with API Connect management system backups.
- **Troubleshooting stanza-create job for S3 backup configuration v10.0.6.0**
  Troubleshoot configuration of s3 backups when the stanza-create job fails.
- **Restore failure with bootstrap pod error v10.0.6.0**
  Postgres bootstrap pod reports an error and restore does not progress.
- **Restore failure with compliance pod error on Kubernetes v10.0.6.0**
  Compliance pod reports an error, and restore does not progress.

Troubleshooting resiliency issues v10.0.6.0

Resolve resiliency issues with API Connect management system backups.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting pgbackrest-shared-repo pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to **`local-storage`**, then pgbackrest-shared-repo pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage: Monitoring Postgres disk usage.

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting stanza-create job for S3 backup configuration v10.0.6.0

Troubleshoot configuration of s3 backups when the stanza-create job fails.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. When you configure Management subsystem backups for s3 providers, the operator runs a **`stanza-create`** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the **`stanza-create`** job:

```
kubectl get jobs -n <namespace> | grep stanza
```

On failure:

- **`stanza-create`** pods fail, and job status is **`0/1`**:

  ```
  kubectl get jobs | grep stanza
   m1-3bfe12ac-postgres-stanza-create          0/1           32m        32m
  ```

- Listing all pods brought up by **`stanza-create`** job shows multiple pods because a job tries **`backoffLimit`** times to complete the job. By default, there will be **`7`** pods in error state:

  ```
  kubectl get pods | grep stanza
   m1-3bfe12ac-postgres-stanza-create-726z4                      0/1     Error       0          11m
   m1-3bfe12ac-postgres-stanza-create-9vq4n                      0/1     Error       0          22m
   m1-3bfe12ac-postgres-stanza-create-gbtgb                      0/1     Error       0          30m
   m1-3bfe12ac-postgres-stanza-create-gl5g5                      0/1     Error       0          17m
   m1-3bfe12ac-postgres-stanza-create-q8qhw                      0/1     Error       0          26m
   m1-3bfe12ac-postgres-stanza-create-t84zs                      0/1     Error       0          8m54s
   m1-3bfe12ac-postgres-stanza-create-z8tfs                      0/1     Error       0          12m
  ```

  Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

  ```
  kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

  time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
  time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
  time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command
  requested"
  time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
  time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest
  stanza-create  --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --
  repo1-type=s3]"
  time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create  --
  db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3 "
  time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
  time="2021-04-13T20:05:21Z" level=info msg="output=[]"
  ```

- Job status is set to **`type: Failed`** and **`status: "True"`**, with **`reason: BackoffLimitExceeded`**:

  ```
   kubectl get job m1-3bfe12ac-postgres-stanza-create -o yaml

  status:
    conditions:
      - lastProbeTime: "2021-04-13T20:28:03Z"
        lastTransitionTime: "2021-04-13T20:28:03Z"
        message: Job has reached the specified backoff limit
        reason: BackoffLimitExceeded
        status: "True"
        type: Failed
        failed: 7
  startTime: "2021-04-13T20:01:18Z"
  ```

- If pod logs are lost, you can execute a command inside the **`backrest-shared-repo`** pod:
  1. Obtain the **`backrest-shared-repo`** pod name:

     ```
     kubectl get pods -n <namespace> | grep backrest-shared-repo
     ```

     For example:

```
kubectl get pods | grep backrest-shared-repo
m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-twl95   1/1     Running
```

2. Exec into the pod:

   ```
   kubectl exec -it <backrest-shared-repo-pod> -- bash
   ```

3. Run:

   ```
   pgbackrest info --output json --repo1-type s3
   ```

   For example, for invalid hostname:

   ```
   ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
   ```

   The exact `ERROR:` will vary depending on the misconfiguration. Common errors:

   Invalid access key
   Invalid access key secret
   Invalid bucket region
   Invalid bucket or folder path

- To fix the incorrect settings, see Reconfiguring or adding backup settings after installation of the management subsystem v10.0.6.0.

Restore failure with bootstrap pod error v10.0.6.0

Postgres bootstrap pod reports an error and restore does not progress.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

Symptoms

The postgres bootstrap pod goes into an error state. Subsequent bootstrap pods come up showing `running`, but the restore does not progress. The logs of the postgres bootstrap pod show:

```
2022-08-01 15:03:58,337 INFO: Lock owner: None; I am management-09384302-postgres-bootstrap-v6xwz
2022-08-01 15:03:58,337 INFO: not healthy enough for leader race
2022-08-01 15:03:58,337 INFO: bootstrap in progress
ERROR: [125]: remote-0 process on 'management-09384302-postgres-backrest-shared-repo.e2e-automation.svc.cluster.local.'
terminated unexpectedly [255]: ssh: connect to host management-09384302-postgres-backrest-shared-repo.e2e-
automation.svc.cluster.local. port 2022: Connection timed out
Mon Aug  1 15:03:58 UTC 2022 ERROR: pgBackRest primary Creation: pgBackRest restore failed when creating primary
2022-08-01 15:03:59,010 INFO: removing initialize key after failed attempt to bootstrap the cluster
2022-08-01 15:03:59,026 INFO: renaming data directory to /pgdata/management-09384302-postgres_2022-08-01-15-03-59
Traceback (most recent call last):
  File "/usr/local/bin/patroni", line 11, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 171, in main
    return patroni_main()
  File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 139, in patroni_main
    abstract_main(Patroni, schema)
  File "/usr/local/lib/python3.6/site-packages/patroni/daemon.py", line 100, in abstract_main
    controller.run()
  File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 109, in run
    super(Patroni, self).run()
  File "/usr/local/lib/python3.6/site-packages/patroni/daemon.py", line 59, in run
    self._run_cycle()
  File "/usr/local/lib/python3.6/site-packages/patroni/__init__.py", line 112, in _run_cycle
    logger.info(self.ha.run_cycle())
  File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1469, in run_cycle
    info = self._run_cycle()
  File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1343, in _run_cycle
    return self.post_bootstrap()
  File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1236, in post_bootstrap
    self.cancel_initialization()
  File "/usr/local/lib/python3.6/site-packages/patroni/ha.py", line 1229, in cancel_initialization
    raise PatroniFatalException('Failed to bootstrap cluster')
patroni.exceptions.PatroniFatalException: 'Failed to bootstrap cluster'
```

Resolution

Delete the Kubernetes API Connect ManagementCluster oplock and restore job, then start the restore again:

```
kubectl delete cm <cr name>-oplock
kubectl delete mgmtr <restoreCR name>
```

Restore failure with compliance pod error on Kubernetes v10.0.6.0

Compliance pod reports an error, and restore does not progress.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. This failure can happen in a situation where a backup is taken on a system where API governance has never been enabled, and the restore of that backup is done onto a system which has API governance enabled. In this situation the API governance database is missing, and consequently the compliance pods report an error.

Symptoms

The compliance service pods are continually restarting, and the restore does not progress. The log of the compliance pod shows the following error:

```
2023-05-19T03:34:31.771Z bhendi:error Database availability check to management-0bfd2507-postgres:5432 failed, try again in
2000 ms,
error: : database "compliance" does not exist, stack: error: database "compliance" does not exist
```

Resolution

Delete all of the compliance jobs, and then start the restore again. For example:

```
kubectl -n <namespace> delete job management-up-compliance-service-data-populate-0-to-1 management-up-compliance-service-
schema-0-to-1
```

Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0

You can back up and restore your Developer Portal service in your Kubernetes environment.

Before you begin

Important: If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.

About this task

To complete a backup and restore of your Developer Portal, you must have the **custom resource** configured with the **portalBackup** subsection. Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Developer Portal if required. Ensure that your backup **path** has the correct permissions to allow your backup user to be able to read and write to it from within the Developer Portal. CR, **custom resource**, templates are provided in the **helper_files** archive to install each API Connect subsystem. You decompressed these files into an installation folder of your choosing in Deploying operators and cert-manager.

Note: For points to consider when you are backing up and restoring a two data center disaster recovery deployment on Kubernetes, see Backup and restore requirements for a 2DCDR deployment.
The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see Overview of the Developer Portal backup resources v10.0.6.0.

Procedure

- If you haven't already, configure your Developer Portal **custom resource** with the **portalBackup** subsection.
  1. Create your Developer Portal backup secret.
     The backup secret is a Kubernetes secret that contains your credentials for accessing the s3 or SFTP backup database.
     - s3
       Only password-based authentication is supported for s3, not authentication based on public certificates and private keys. Password-based authentication for s3 requires that you generate an access key and secret. For example:
       - IBM (Cloud Object Storage): Service credentials.
       - AWS: Managing access keys.

       The secret can be created with the following command:

       ```
       kubectl create secret generic portal-backup-secret --from-literal=username='<YOUR ACCESS KEY OR USER NAME>'
           --from-literal=password='<YOUR ACCESS KEY SECRET OR PASSWORD>' -n <namespace-of-portal-subsystem>
       ```

     - SFTP
       Supported credentials types:
       - Username and password (v10.0.2.0 or later).
       - Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.[1]

       Use one of the following commands to create the secret:
       - Username and password credentials

         ```
         $ kubectl create secret generic portal-backup-secret --from-literal=username='<YOUR USERNAME>'
         --from-literal=password='<YOUR PASSWORD>' -n <namespace-of-portal-subsystem>
         ```

       - Version 10.0.3.0 or later: Username and SSH-key credentials:

```
$ kubectl create secret generic portal-backup-secret --from-literal=username='<YOUR USERNAME>'
 --from-file=ssh-privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-portal-subsystem>
```

2. Ensure that your Developer Portal **custom resource** is configured with the **portalBackup** subsection.
   - All of your settings must be added in the **spec:** section of the file.
   - To schedule automated backups, specify values for the **schedule** setting.

   An example when you use **protocol: objstore**.

```
spec:
  portalBackup:
    credentials: portal-backup-secret
    host: s3.eu-gb.cloud-object-storage.appdomain.cloud/
    path: test-bucket/restore-test
    port: 443
    protocol: objstore
    backupCerts: <custom-s3-server-CA-cert>
    backups3URIStyle: <host-or-path>
    schedule: '0 2 * * *'
```

   An example when you use **protocol: sftp**.

```
spec:
  portalBackup:
    credentials:  portal-backup-secret
    host:         sftp-service.example.com
    path:         /home/fvtuser/site-backups
    port:         22
    protocol:     sftp
    schedule:     '0 2 * * *'
```

   Notes about **portalBackup:** entry:

   - If **portalBackup:** entry is not present or contain nothing, just **local** backups will be taken if a backup CR is created. If the **schedule** exists, backups are taken at the scheduled time.
   - For S3 and SFTP, **portalBackup:** entry must contain host, credentials, protocol, and path.
   - For s3, the **portalBackup:** section can optionally also contain either, or both, of **backupCerts** and **backupS3URIStyle**.

Table 1. Developer Portal backup settings

Setting	Description

credentials	The name of the secret you created.
	Supported credential types:
	• s3
	Only password-based authentication is supported for s3, not authentication based on public certificates and private keys. Password-based authentication for s3 requires that you generate an access key and secret.
	• SFTP
	  • Username and password (v10.0.2.0 or later)
	  • Username and SSH-key (v10.0.3.0 or later)
	To create a secret, see step 1.

host	• s3 - **host**: is the S3 endpoint with the corresponding S3 region in the format **<S3endpoint>**, or with an optional S3 region in the format **<S3endpoint>/<S3region>**. • sftp - the hostname of the sftp server
path	• S3 - the name of your S3 bucket to store backup data, optionally followed by a **/** and then any subdirectories inside the bucket. Valid examples: **bucket** **bucket/path** **bucket/lots/of/paths** • sftp - the full absolute path of the folder on the SFTP server, beginning with **/**.
port	The port for the protocol to connect to the host. • S3 - Defaults to 443 if not specified. • SFTP - Defaults to 22 if not specified. • local - Ignored for local.
protocol	The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values: • **sftp** - for secure file transfer protocol. • **objstore** - for S3 compatible object storage.

Setting	Description

`backupCerts`	(S3 storage only) `backupCerts` is the name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2.
	Note: If you use a public certificate for the S3 storage provider, it must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload:
	`x509:`
	`certificate signed by unknown authority`
	This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be `ca.crt` and the value is the base64-encoded value of the CA certificate.
	If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate.
	If you do need to provide the CA certificate then you must provide the entire chain in the `ca.crt` member of the secret, as follows:
	`intermediate-certificate-1`
	`intermediate-certificate-2`
	`. . .`
	`intermediate-certificate-n`
	`root-certificate`
	Where the first certificate in the `ca.crt` member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the `ca.crt` member is the issuer of the preceding certificate. The root certificate in the `ca.crt` member must be self-signed.
	If there are no intermediate certificates involved, then the `ca.crt` member contains only the root certificate.
	The following sample bash script creates the Kubernetes secret:
	`cat >customs3ca.yaml <<EOF`
	`apiVersion: v1`
	`data:`
	`ca.crt: $(base64 <path-to-ca-certificate> | tr -d '\n' )`
	`kind: Secret`
	`metadata:`
	`name: custom-server-ca`
	`type: generic`
	`EOF`
	`kubectl apply -f customs3ca.yaml -n <namespace>`

`backups3URIStyle`	(S3 storage only) Optional - `backups3URIStyle` indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, default to `host`. Supported beginning with 10.0.2. Valid values: `host` and `path`. Some custom S3 providers require URI style to be set to `path`. For example, MinIO supports both `host` and `path` style setup. You can create a MinIO S3 server to only accept `path` style client communications. **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.

`schedule`	The schedule for how often automatic Developer Portal backups are run. The format for the schedule is any valid cron string. The timezone for backups is UTC.
	Valid for all backup types: s3, sftp, and local.
	`spec:`
	`portalBackup:`
	`...`
	`schedule: '0 2 * * *'`

`backupRecordDays`	The `backupRecordDays` setting denotes the number of days that backup records (backups that exist on the remote backup server) are shown when you list your remote backups. It is defaulted to 30 days.

- Reconfiguring Developer Portal backups
  You can edit the deployed Developer Portal `custom resource` You might want to do an edit if you did not include a `portalBackup` section before the installation. Or, if you made a mistake in your `portalBackup`, or if you'd like to change the `portalBackup` config. To edit your `portalcluster` CR:
  1. Get the name of your `portalcluster` CR:

     `kubectl get portalcluster -n <namespace-of-portal-subsystem>`

  2. Edit your `portalcluster` CR:

     `kubectl edit portalcluster <name-of-portal-cr> -n <namespace-of-portal-subsystem>`

  3. Inside the `spec`, add the `portalBackup` section as explained in step 2.
  4. Write and quit **wq**.
- How to generate a manual backup.
  You can use these steps to configure a Developer Portal with backups that use IBM's Cloud Object Storage S3, AWS S3, or SFTP.

  Decide which of the three types of manual backups you want to do:
  - `all`: backs up the Developer Portal system and all its sites to the remote server
  - `system`: backs up the Developer Portal system to the remote server
  - `site`: backs up a Developer Portal site to the remote server

  1. You generate a backup manually by creating a `portalBackup` `custom resource` that is detected by the `ibm-apiconnect` operator that then triggers the backup process.
     You can use portalbackup_cr.yaml in the helper_files directory as an example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  generateName: portal-bup-
spec:
  type: all
  portalCluster: portal
  crType: create
  comment: "test comment"
```

- `type`: is the type of backup to be done. Valid arguments are:
  - `all`
  - `system`
  - `site`
- `portalCluster`: The name of the portal cluster you want to backup. You can find this with:

  `kubectl get portalcluster -n <namespace>`

- `crType`: Use `create` to trigger a backup process. When the backup is created, the operator generates another backup CR with `crType: record`, which represents the stored backup that you can use for a restore operation.
- `comment`: you can optionally add a comment to the CR before you create a backup.
- `siteName`: used only when you take a site backup. Valid arguments are:
  - site uuid or url - back up that site only to the remote server
  - `installed` - backups all installed sites to the remote server

2. Create the `PortalBackup custom resource` in the `namespace` of your Developer Portal to trigger the backup process:

   `kubectl create -f portalbackup_cr.yaml -n <ns-of-portal-subsystem>`

3. Optional: You can list current backups by using the command:

   `kubectl get portalbackup -n <ns-of-portal-subsystem>`

   Here is an example of some possible output:

```
NAME               ID             STATUS     TYPE     CR TYPE   AGE     COMMENT
p1-bup-gqz8f    20200526.142449  Ready      system   record    55m
p1-bup-jx4vc    20200526.142458  Ready      site     record    54m
p1-bup-st412                     Ready      all      create    56m     y
p1-bup-th2dm                     Ready      all      create    4h51m   y
```

   Tip: If you want to order your `portalbackups` by ID, you can use the Kubernetes keyword `--sort-by`. For example:

   `kubectl get pb -n <ns> --sort-by=.status.backupId`

4. Optional: You can get a detailed view of a specific backup by using the command:

   `kubectl get portalbackup <bup-name> -n <ns-of-portal-subsystem> -o yaml`

   Here is an example of some possible output:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-
20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-
storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
```

```
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnnjnu
  message: ""
  phase: Ready
  state: ""
```

- How to list available Developer Portal backups on remote server.
  Use command:

  ```
  kubectl get portalbackups -n <ns-of-portal-subsystem>
  ```

  Here is an example of some possible output:

  ```
  kubectl get portalbackup -n <ns-of-portal-subsystem>
    NAME               ID             STATUS     TYPE     CR TYPE   AGE     COMMENT
    p1-bup-gqz8f    20200526.142449  Ready      system   record    55m
    p1-bup-jx4vc    20200526.142458  Ready      site     record    54m
    p1-bup-st4l2                     Ready      all      create    56m     y
    p1-bup-th2dm                     Ready      all      create    4h51m   y
  ```

  Note: You can only restore the Portal subsystem from backups where the `CR TYPE` is `record`.
  Optionally, if you need more information, such as the name of the .tgz file that you can run, use command:

  ```
  kubectl get portalbackups <bup-name> -n <ns-of-portal-subsystem> -o yaml
  ```

  Here is an example of some possible output:

  ```
  apiVersion: portal.apiconnect.ibm.com/v1beta1
  kind: PortalBackup
  metadata:
    creationTimestamp: "2020-10-01T10:01:19Z"
    generateName: portal-bup-
    generation: 1
    name: portal-bup-z7wgk
    namespace: default
    resourceVersion: "8938"
    selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-z7wgk
    uid: 5c5899c0-4da0-4468-b7fb-1c283121a053
  spec:
    comment: ""
    portalCluster: portal
    crType: record
    siteName: ""
    type: site
  status:
    backupId: "20201001.100021"
    commentLeft: ""
    conditions:
    - lastTransitionTime: "2020-10-01T10:01:19Z"
      reason: BackupFailed
      status: "False"
      type: Error
    - lastTransitionTime: "2020-10-01T10:01:19Z"
      message: Record Backup CR for conor.com-20201001.100021.tar.gz
      reason: BackupComplete
      status: "True"
      type: Ready
    - lastTransitionTime: "2020-10-01T10:01:19Z"
      reason: BackupInProgress
      status: "False"
      type: Running
    - lastTransitionTime: "2020-10-01T10:01:19Z"
      reason: BackupStatusUnknown
      status: "False"
      type: Warning
    fullBackupName: conor.com-20201001.100021.tar.gz
    id: ""
    message: ""
    phase: Ready
    state: ""
  ```

- How to restore a Developer Portal subsystem.
  Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Developer Portal subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore.

  There are three types of restores:
  - `all`: restores the Developer Portal system and all the sites that are found on the remote server by using the latest backup.
  - `system`: restores the Developer Portal system to the remote server
  - `site`: restores the Developer Portal site from the remote server

  1. Decide on the options to use for your `PortalRestore custom resource`. The `PortalRestore custom resource` is detected by the `ibm-apiconnect`operator that then triggers the backup process.
     You can use portalrestore_cr.yaml in the helper_files directory as an example:

     ```
     apiVersion: portal.apiconnect.ibm.com/v1beta1
     kind: PortalRestore
     metadata:
       generateName: portal-restore-
     spec:
     ```

```
  type: all
  portalCluster: portal
  dryRun: true
  timestamp: now
  priorityList:
    - myportal-critical-site.com
    - myportal-important.com
```

- **type**: is the type of restore to be done. Valid arguments are:
  - **all**
  - **system**
  - **site**
- **portalCluster** is the name of the portal cluster you want to restore. You can find this with:

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

- **siteName**: used only when you took a site backup. Valid arguments are:
  - *ID*, for example 20200526.142458: the backup ID of a portal backup CR. The ID must be taken from a portal backup that is of **type** **site**, and **crType** **record**.
  - site-1234.tgz: restore the site by using this file name that exists on the local file system or remote server.
  - myportal.com: find the latest backup for this URL (locally or remotely) and restore it.
  - **all**: restore all sites that are found on the remote server by using the latest backup.
  - **site** uuid or url - back up that site only to the remote server
  - **installed** - backups all installed sites to the remote server.
- **systemName** - Valid arguments are:
  - *ID*, for example 20200526.142449: the backup ID of a portal backup CR. The ID must be taken from a portal backup that is of **type** **system**, and **crType** **record**.
  - system_backup-1234.tgz: restores the system from the given file that exists on the pod's local file system or the remote server.
  - **latest**: restores from latest system backup that is found from either local file system or remote server.
- **timestamp** - Valid arguments are:
  - **now**: use the latest backups available.
  - < TIMESTAMP >: in ***'YYYYMMDD.HHMMSS'*** format, specify a timestamp to retrieve the backup from. The nearest backup, searching backwards from this timestamp, is used.

  Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to ***YYYYMMDD.HHMMSS***. For Version 10.0.2.0 and earlier, the timestamp format is ***YYYY-MM-DD HH:MM:SS***.
- **dryRun** - Valid arguments are:
  - **true**: runs a dry run of the command, returning the actions that will be taken. These actions can be found in the custom resource by following step 4.
  - **false**: runs the command, restoring and replacing the system files and all sites. Any existing sites are reinstalled.
- **priorityList** - Prioritizes the restore of any sites. The sites are listed in the order in which they are provided. They are queued for restore first, followed by any remaining sites present in the backup location. Valid arguments are:
  - **myportalsite.com**
  - **mysecondportalsite.com**
- **customPlatformApiHostname** - If the **API** hostname of your management subsystem platform has changed since taking the backup you are restoring, then update it to the new platform API hostname of the management subsystem. (Optional)

Table 2. The options that are needed for each Developer Portal restore

	all	system	site
type	X	X	X
portalCluster	X	X	X
siteName			X
systemName		X	
timestamp	X		
dryRun	X		
priorityList	X		

2. Create the **PortalRestore** **custom resource** in the **namespace** of the Developer Portal subsystem to trigger the restore process:

   ```
   kubectl create -f portalrestore_cr.yaml -n <ns-of-portal-susbystem>
   ```

3. Optional: You can list current restores by using the command:

   ```
   kubectl get portalrestore -n <ns-of-portal-susbystem>
   ```

   Here is an example of some possible output:

   ```
   NAME                   READY     AGE
   portal-restore-twjrp   Ready     28s
   ```

4. Optional: You can get a detailed view of a specific restore by using the command:

   ```
   kubectl get portalrestore <restore-name> -n <ns-of-portal-susbystem> -o yaml
   ```

   Here is an example of some possible output:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalRestore
   metadata:
     creationTimestamp: "2020-11-11T10:14:39Z"
     generateName: portal-restore-
     generation: 1
     name: portal-restore-8f4c5
     namespace: portal
   ```

```
  resourceVersion: "1497597"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-8f4c5
  uid: 5a70b657-d998-4989-b9b6-8480e986311d
spec:
  siteName: "20200526.090209"
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestorePending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    message: |-
      2020-05-26 13:11:25: CLI task (restore:site) starting.
      2020-05-26 13:11:36: Deleting any old site data if it exists...
      2020-05-26 13:13:18: Restoring site test.com ...
      2020-05-26 13:13:18: .
      2020-05-26 13:13:19: Patching the database dump before restoring ...
      2020-05-26 13:13:48: .
      2020-05-26 13:19:41: Triggering the drush command to restore the site. This may take some time.
      2020-05-26 13:19:48: .
      2020-05-26 13:25:40: Drush restore command completed. Setting up the site.
      2020-05-26 13:25:48: .
      2020-05-26 13:30:49: Site test.com restored from backup /var/aegir/backups/test.com-20200526.090209.tar.gz.
Initializing the site...
      2020-05-26 13:30:52: Restore site completed.
      2020-05-26 13:30:52: CLI task (restore:site) completed successfully.
    reason: RestoreComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    reason: RestoreInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: bybyibd99wxtmkmk
  message: ""
```

- Troubleshooting
  1. Check the Developer Portal backup CR or restore CR.
     Use command:

     ```
     kubectl get <portalbackup/portalrestore> <bup-name/restore-name> -n <namespace> -o yaml
     ```

     Here is an example of some possible output:

     ```
     apiVersion: portal.apiconnect.ibm.com/v1beta1
     kind: PortalRestore
     metadata:
       creationTimestamp: "2020-05-26T13:10:46Z"
       generateName: portal-restore-
       generation: 1
       name: portal-restore-x2bf9
       namespace: portal
       resourceVersion: "1493471"
       selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-x2bf9
       uid: 9f6a1e22-55ab-4595-8b98-bdfaec40d58c
     spec:
       portalCluster: portal
       type: site
     status:
       backupId: ""
       commentLeft: ""
       conditions:
       - lastTransitionTime: "2020-05-26T13:10:49Z"
         message: |
           Please provide a site to restore. To restore all sites use value 'all' in the CR
         status: Failed
         type: Ready
       fullBackupName: ""
       id: ""
       message: ""
     ```

     Read the message inside the conditions of the Developer Portal backup CR or the restore CR.

  2. Retrieving logs
     Look for information in the logs of the `ibm-apiconnect` pod:
     - Locate the `ibm-apiconnect` pod with:

       ```
       kubectl get pods -n <namespace-of-portal-subsystem>
       ```

     - View the logs with:

```
kubectl logs <ibm-apiconnect-pod-NAME> -n <namespace-of-portal-subsystem>
```

- **Overview of the Developer Portal backup resources v10.0.6.0**
  How to manage your Developer Portal remote backup resources.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Overview of the Developer Portal backup resources v10.0.6.0

How to manage your Developer Portal remote backup resources.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Note: This information is relevant only if you're taking remote Developer Portal backups.
The Developer Portal uses two types of portal backup resources. When you run the command **`kubectl get portalbackup`**, you see a list of backups, for example:

```
NAME                  ID                TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE     COMMENT
my-backup-f91ms       n/a               20b47fe090f5 Ready    system   create    portal    78m
portal-record-222cg   20220108.010033                Ready    system   record    portal    94m
portal-record-22xdc   20220702.010339                Ready    site     record    portal    9m49s
portal-record-245j7   20221002.010340                Ready    site     record    portal    5m
portal-record-246hm   20221218.010103                Ready    site     record    portal    32m
portal-record-24h44   20220509.010037                Ready    site     record    portal    68m
portal-record-24wb4   20221003.010029                Ready    site     record    portal    60m
portal-record-264mh   20221119.010453                Ready    system   record    portal    78m
```

There are two types of resources listed, and these are denoted by the **`CR TYPE`** column; **`create`** or **`record`** types.

1. **`create`**: This resource is a representation of a **`portalbackup`** task that was created manually. See Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0, Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0, or Backing up the Developer Portal subsystem on VMware, for information about how to create a manual portal backup.
2. **`record`**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

   ```
   kubectl describe portalbackup portal-record-name
   ```

   For example:

   ```
   kubectl describe portalbackup portal-record-zmmfw
   .
   .
   Name:          portal-record-zmmfw
   Namespace:     apic
   Labels:        createdBy
   .
   .
   Full Backup Name:         my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
   ```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up and restoring the analytics database

The API Connect analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

For information about analytics backup, restore, and disaster recover, refer to Backing up and restoring the analytics subsystem.

Backups on OpenShift V10.0.6.0

You can backup and restore API Connect subsystems on OpenShift and Cloud Pak for Integration.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- **Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can use the following instructions to backup and restore your Management subsystem on OpenShift and Cloud Pak for Integration.
- **Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can back up and restore your Developer Portal service in your OpenShift or Cloud Pak for Integration environment.
- **Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration**
  The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0

You can use the following instructions to backup and restore your Management subsystem on OpenShift and Cloud Pak for Integration.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- To complete a backup and restore of your Management subsystem, you must have the `custom resource` configured with the `databaseBackup` subsection in the management subsystem custom resource.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required.
- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.
- In v10.0.1.1-eus and later, a new parameter called `protocol` is required in `databaseBackup` section
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.

Note: You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported

- Cloud-based S3 storage
  - IBM Cloud Object Storage
  - AWS S3
- SFTP
  - In v10.0.1.1-eus, SFTP backup/restore procedure is automatically handled by apiconnect operator.
  - In v10.0.1.0 and later, local Backups can be stored remotely on an sftp server, and retrieved later from the sftp server for restore operations. See Using an sftp server for backup files (v10.0.1.0).
- Local storage backups

Support for Cloud Pak for Integration

IBM Cloud Pak for Integration is deployed on OpenShift, and uses the API Connect distribution for OpenShift. The backup and restore instructions in this section apply to both API Connect on OpenShift, and API Connect within a Cloud Pak for Integration environment.

When you perform configuration tasks you can use either a graphical user interface or a command line. The IBM Cloud Pak Platform UI differs from the OpenShift web console. The configuration topics in this section provide instructions for both interfaces. Note that Cloud Pak for Integration uses the OpenShift command line utility `oc`. Command line instructions for configuration are identical for both platforms.

- **Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0**
  You can configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment. Use the appropriate instructions for the type of backup that you want to configure:
- **Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can reconfigure the backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration.
- **Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can generate a manual (non-scheduled) backup of the management subsystem in your OpenShift and Cloud Pak for Integration environments.
- **Restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can restore your Management subsystem in your OpenShift and Cloud Pak for Integration environments.
- **Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can troubleshoot failures of the API Connect management subsystem backups in an OpenShift or Cloud Pak for Integration environment.

Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0

You can configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment. Use the appropriate instructions for the type of backup that you want to configure:

- **Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0**
  Configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.
- **Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Beginning with API Connect V10.0.1.1, you can configure SFTP backups for your Management subsystem in your OpenShift and Cloud Pak for Integration environments.
- **Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0**
  Configure local backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0

Configure backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

Before you begin

If the **APIConnectCluster** instance is already created, do not use these instructions. Instead, add the Management backup configuration by following the instructions in Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem. Beginning with v10.0.2, the Management subsystem supports custom S3 solutions such as MinIO.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions that can cause database restores to fail.
Warning:
When configuring a new management subsystem for S3 backup, the content of the S3 bucket path must be empty. If you want to configure a management subsystem to use the content of an S3 bucket path already used by another management subsystem, you have two options:

1. Shutdown the postgres database on the existing management subsystem that points to the S3 bucket path before you start the new management subsystem.
2. In a disaster recovery scenario where you want to restore from the backup of an existing installation: Create a new S3 path and use S3 tools to copy the backup content from the existing management subsystem S3 bucket path, to the new bucket path.

Procedure

1. Create a backup secret.
   The backup secret is a Kubernetes secret that contains your username and password for your S3 backup database.

   For examples of how to generate the appropriate access key and secret for two of them, see:

   - IBM (Cloud Object Storage): Service credentials.
   - AWS: Managing access keys.

   The secret can be created with the following command:

   ```
   oc create secret generic mgmt-backup-secret --from-literal=key='<YOUR ACCESS KEY>' --from-literal=keysecret='<YOUR ACCESS KEY SECRET>' -n <namespace_of_management_subsystem>
   ```

2. Configure the Management backup settings.
   Create the **APIConnectCluster** installation CR and add the Management backup configuration using one of the following methods:

   - OpenShift web console:
     - Navigate to Advanced Configuration > ManagementSubsystem > Advanced Configuration > Database Backups > Advanced Configuration
     - See the on-screen instructions for how to fill in each field. See also Management subsystem settings.
       Note that there is downtime involved when changing S3 protocol backup configuration.

   - IBM Cloud Pak Platform UI:
     - In the Platform UI for the API Connect cluster instance, select Configuration. On the Form UI tab, set Advanced Options to On.
     - See the on-screen instructions for how to fill in values for:
       - Credentials
       - Hostname
       - Path
       - Protocol
       - Restart DB (toggle on)
       - S3 Provider
       - Backup Credentials
       - Backup URI Style
       - Schedule (The timezone for backups is that of the node on which the postgres-operator pod is scheduled.)
       - Repo Retention

     For more information, see Management subsystem settings.

   - YAML file: Use Table 1 for guidance while you define the backup settings as shown in the following example.

     ```
     spec:
       management:
         databaseBackup:
           protocol: objstore
           s3provider: ibm
           host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
           path: apic-backup
           credentials: mgmt-backup-secret
           schedule: "0 3 * * *"
           repoRetentionFull: 2
     ```

     Note: The **protocol** setting was introduced in v10.0.1.1. Do not specify it on v10.0.1.0 or earlier deployments

Table 1. Backup configuration settings

Setting	Description
protocol	The type of the backup; required in 10.0.1.1-eus and greater. For s3 storage, use **objstore**. When the parameter is not set, the backup type defaults to local storage backup. Note: The **protocol** setting was introduced in v10.0.1.1. Do not specify it on v10.0.1.0 or earlier deployments
s3Provider	Name of the S3 provider to use. You must specify one of the supported values of **aws**, **ibm**, or **custom**. The **custom** option is supported beginning with 10.0.2 or later. If you use a custom S3 provider, you must also specify the **backupCerts**. Note: The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: **x509: certificate signed by unknown authority**.

Setting	Description
backupCerts	The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. Used only when **s3Provider** is set to **custom**. Most of the custom S3 providers are created based on custom CA certificates. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be **ca.crt** and the value is the base64-encoded value of the CA certificate. If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Analytics subsystem already trusts the server certificate. You can test this by configuring the Analytics backup to your S3/object-store server without providing the CA certificate. If you do need to provide the CA certificate then you must provide the entire chain in the **ca.crt** member of the secret, as follows: **intermediate-certificate-1** **intermediate-certificate-2** **. . .** **intermediate-certificate-*n*** **root-certificate** Where the first certificate in the **ca.crt** member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the **ca.crt** member is the issuer of the preceding certificate. The root certificate in the **ca.crt** member must be self-signed. If there are no intermediate certificates involved, then the **ca.crt** member contains only the root certificate. The following sample bash script creates the Kubernetes secret: **cat >customs3ca.yaml <<EOF** **apiVersion: v1** **data:** **ca.crt: $(base64 *<path-to-ca-certificate>* | tr -d '\n')** **kind: Secret** **metadata:** **name: custom-server-ca** **type: generic** **EOF** **kubectl apply -f customs3ca.yaml -n *<namespace>*** **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.
backups3URIStyle	Indicates whether URIs to your S3 backup should use specify a host or a path value. Supported beginning with 10.0.2. Valid values: **host** and **path**. Used only when **s3Provider** is set to **custom**. Some custom S3 providers require URI style to be set to **path**. For example, MinIO supports both **host** and **path** style setup. You can create a MinIO S3 server to only accept **path** style client communications. **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.
host	For **objstore** type backup, specify S3 endpoint with the corresponding S3 region in the format **<S3endpoint>/<S3region>**
path	The path to the location of the backup: For **objstore** backups, the name of your S3 bucket to store backup data. For example, **bucket-name/folder**. The use of subdirectories in the bucket name is not supported. Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name. Ensure that **bucket-name/folder** is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. **Explanation:** Once a folder is used, archive.info and backup.info files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring.
credentials	For **objstore** type backups, the name of the Kubernetes secret containing your S3 Access Key / Key Secret
schedule	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------ min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. There is no default backup schedule set. Be sure to set your backup schedule. All scheduled Management subsystem backups are of type **full** only.

Setting	Description
`repoRetentionFull`	**v10.0.3.0 or later** - The number of full S3 backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: `1` Maximum value: `9999999` Default: none

3. Verify that the configuration deploys successfully.
   When you configure Management subsystem backups, the operator runs a `stanza-create` job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

   Check the status of the `stanza-create` job:

   ```
   oc get jobs -n <namespace> | grep stanza
   ```

   - On success:
     - The `stanza-create` job status is `1/1`:

       ```
       oc get jobs | grep stanza
       m1-b722e361-postgres-stanza-create          1/1           5s         127m
       ```

     - The `stanza-create` job shows `status: "True"` and `type: Complete`:

       ```
       oc get job m1-b722e361-postgres-stanza-create -o yaml

       status:
         completionTime: "2021-04-13T18:38:50Z"
         conditions:
           - lastProbeTime: "2021-04-13T18:38:50Z"
             lastTransitionTime: "2021-04-13T18:38:50Z"
             status: "True"
             type: Complete
        startTime: "2021-04-13T18:38:45Z"
        succeeded: 1
       ```

     - The `stanza-create` pod is in `Completed` state:

       ```
       oc get pods | grep stanza
       m1-b722e361-postgres-stanza-create-q4fvp                      0/1     Completed   0          127m
       ```

     - Exec into the pod:

       ```
       oc exec -it <backrest-shared-repo-pod> -- bash
       ```

     - The `pgbackrest` command returns the S3 contents in valid JSON:

       ```
       pgbackrest info --output json --repo1-type s3

       "backup":{"held":false}},"message":"ok"}][pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdfdd5-rs882
       /
       ```

   - On failure, see Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration v10.0.6.0.

Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0

Beginning with API Connect V10.0.1.1, you can configure SFTP backups for your Management subsystem in your OpenShift and Cloud Pak for Integration environments.

Before you begin

If the `APIConnectCluster` instance is already created, do not use these instructions. Instead, add the Management backup configuration by following the instructions in Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem.

Procedure

1. Create a backup secret.
   The backup secret is a Kubernetes secret that contains your credentials for accessing the SFTP backup database. Supported credentials types:

   - Username and password (v10.0.2.0 or later).
   - Username and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.[1]

   Use one of the following commands to create the secret:

   - Username and password credentials

```
oc create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-literal=password='<YOUR PASSWORD>' -n <namespace-of-mgmt-subsystem>
```

- **Version 10.0.3.0 or later**: Username and SSH-key credentials:

```
oc create secret generic mgmt-backup-secret --from-literal=username='<YOUR USERNAME>' --from-file=ssh-privatekey='<YOUR PRIVATEKEY FILE>' -n <namespace-of-mgmt-subsystem>
```

Note: If the username contains the backslash character, insert an extra backslash to escape it. For example, if the username is:

```
backup\svc-apic
```

set this as:

```
-from-literal=username='backup\\svc-apic'
```

2. Configure the Management backup settings.
Create the `APIConnectCluster` installation CR and add the Management backup configuration using one of the following methods:

- OpenShift web console:
  - Navigate to Advanced Configuration > ManagementSubsystem > Advanced Configuration > Database Backups > Advanced Configuration
  - See the on-screen instructions for how to fill in each field. See also Management subsystem settings.
    The following fields are unique to SFTP backup configuration:
    - Server port. Default is 22.
    - The fields in the Restart Database section. Changing SFTP (protocol) backup configures does not involve any downtime.
    - Retries.
- IBM Cloud Pak Platform UI:
  - In the Platform UI for the API Connect cluster instance, select Configuration. On the Common settings tab, set Advanced Options to On.
  - See the on-screen instructions for how to fill in values for:
    - Credentials
    - Server Hostname
    - Path
    - Server Port
    - Protocol
    - Retries
    - Schedule (The timezone for backups is that of the node on which the postgres-operator pod is scheduled.)

    For more information, see Management subsystem settings.
- YAML file: Use Table 1 for guidance while you define the backup settings as shown in the following example.

```
spec:
  management:
    databaseBackup:
      protocol: sftp
      host: <SFTP-host-name>
      port: <SFTP-port>
      path: apic-backup
      retries: 0
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
```

Table 1. Backup configuration settings

Setting	Description
`protocol`	The type of the backup. For SFTP storage: `sftp`.
`host`	The backups host. For `sftp` type, the SFTP server hostname
`port`	The SFTP server port. Optional. Default: `22`.
`path`	The path to the location of the backup. For `sftp` type, the folder name on the SFTP server.
`retries`	The number of times the `ibm-apiconnect` Operator attempts backups in the event of a failed SFTP backup. Default value: `0`.
`credentials`	Name of the Kubernetes secret containing your SFTP Username/password. For Version 10.0.3.0 or later, the Kubernetes secret can contain your SFTP Username and SSH key.
`schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. There is no default backup schedule set. Be sure to set your backup schedule. All scheduled Management subsystem backups are of type `full` only.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Configuring local backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0

Configure local backups for the API Connect Management subsystem that you will deploy in your OpenShift or Cloud Pak for Integration environment.

About this task

When creating an instance of API Connect, use the advanced configuration settings to configure database backups for the Management subsystem.

Procedure

Configure the Management section of the top-level `APIConnectCluster` installation CR with the `databaseBackup` subsection.
Specify `schedule` and `repoRetentionFull`, the default settings are 01:00 UTC and 14 days.

```
databaseBackup:
  schedule: "0 1 * * *"
  repoRetentionFull: 14
```

Table 1. Backup configuration settings

Setting	Description
`schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The timezone for backups is that of the node on which the postgres-operator pod is scheduled. The default is daily backups at 01:00 UTC. All scheduled Management subsystem backups are of type `full` only.
`repoRetentionFull`	- The number of full local backups to retain. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: `1` Maximum value: `9999999` Default: 14

Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0

You can reconfigure the backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration.

About this task

Backup configurations under `management.databaseBackup` in `APIConnectCluster` instance can be modified using the OpenShift web console, IBM Cloud Pak Platform UI, or the command-line utility `oc`.

Note:
Using OpenShift web console or Platform UI can prevent updates because `APIConnectCluster` CR refreshes very quickly. If you encounter this problem, use the command line to modify backup configurations; for example:

```
oc edit apiconnectcluster <name-of-instance>
```

Procedure

1. Make sure that the management cluster status is `Running` and the READY condition displays the same value before and after the "/".
   For example:

   ```
   oc get mgmt
   NAME    READY   STATUS    VERSION      RECONCILED VERSION   AGE
   mgmt    16/16   Running   10.0.1-eus   10.0.1.1-eus-0-eus       19h
   ```

2. Create backup secret for your backup type. See the appropriate instructions:
   - Configuring S3 backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0
   - Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0
     Note that SFTP backups settings are not supported on v10.0.1.0 or earlier.

3. Complete the following steps, based on your storage type:
   - S3 storage

If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites. Follow these steps:

- Remove the **multiSiteHA** section from the **spec** section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the **multiSiteHA** sections that you remove.
- Configure your S3 backup settings. You must specify different backup locations for each data center, see Backup and restore requirements for a 2DCDR deployment.
- Add the **multiSiteHA** sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

You can change or add the **DatabaseBackup** s3 configuration via the Management section of the API Connect Cluster CR after a fresh install. However, doing so requires a reconfiguration of the database and a brief downtime. The operator restarts the database with the new backup configurations as soon as the new configuration in the management CR is saved.

```
spec:
  management:
    databaseBackup:
      protocol: objstore
      s3provider: ibm
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: apic-backup
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
      restartDB:
        accept: true
```

Configuration of s3 backups uses a new parameter **restartDB** that is required when changing the backup configuration. **restartDB** is not required for initial deployments, only for modifications after the cluster is deployed.

Table 1. Additional parameter for S3 backup type reconfiguration

Parameter	Description
`restartDB:` `accept: <value>`	Required when changing the backup configuration. The parameter **accept** is a boolen: <*value*> can be **true** or **false**. When **accept** is **true**, and a new configuration for management CR is saved, the operator: • Stops the database and any dependent services • Applies the change • Restarts the stopped components. Note: **restartDB** is not required for initial deployments, only for modifications after the cluster is deployed.

- SFTP

  You can change/add the **DatabaseBackup** SFTP configuration via the Management section of the API Connect Cluster CR after a fresh install. This process does not involve any downtime. The same backup settings are used for reconfiguring as for initial configuration. For complete description of backup settings, see Configuring SFTP backup settings for fresh install of the Management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

```
spec:
  management:
    databaseBackup:
      protocol: sftp
      host: <SFTP-hostname>
      port: <SFTP-port> (Optional, default is 22)
      path: apic-backup
      retries: 0
      credentials: mgmt-backup-secret
      schedule: "0 3 * * *"
```

Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0

You can generate a manual (non-scheduled) backup of the management subsystem in your OpenShift and Cloud Pak for Integration environments.

Procedure

1. Make sure that the management cluster status is Running and the READY condition displays matching values before and after the "\".
   You can use either the OpenShift web console or the **oc** command line to verify the health. For example:

   ```
   oc get mgmt
   NAME   READY   STATUS    VERSION      RECONCILED VERSION   AGE
   mgmt   16/16   Running   10.0.1-eus   10.0.1.1-eus-0-eus   19h
   ```

2. Decide which of the two types of manual backups you want to do:
   - **full**: backs up the entire Management Subsystem database.
   - **incr**: backs up any data that has not been backed up since the last **full** or **incr** backup.
3. You generate a backup manually by creating a managementBackup custom resource which is detected by the ibm-apiconnect operator that then triggers the backup process.
   You can create the **ManagementBackup** custom resource by using either the Form UI or the command line. For example:

   ```
   apiVersion: management.apiconnect.ibm.com/v1beta1
   kind: ManagementBackup
   metadata:
     generateName: mgmt
   spec:
     type: full
     crType: create
     clusterName: management
   ```

Table 1. CR settings for manual backup

Setting	Description
`type`	The type of backup to be done. Valid arguments are: • `full` Backs up the entire Management subsystem database • `incr` Backs up any data that has not been backed up since the last `full` or `incr` backup.
`crType`	Use `create` to trigger a backup process.
`clusterName`	The name of the target ManagementCluster you want to backup.

4. Create the ManagementBackup custom resource in the namespace of your Management Subsystem to trigger the backup process. Use the Form UI or the following command:

```
$ oc create -f mgmtbackup_cr.yaml -n <APIC_namespace>
```

5. (Optional): You can list current backups by using Form UI to list `ManagementBackup` resources, or by using `oc`:

```
$ oc get managementbackup -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
```

Here is an example of some possible output:

```
NAME                STATUS   ID                 CLUSTER   TYPE   CR TYPE   AGE
m1-e46bbac4         Ready    20201023-183005F   m1        full   record    66m
mgmt-backup-dnvxw   Ready    20201023-193137F   m1        full   create    9m37s
```

6. (Optional): You can get a detailed view of a specific backup by using the command:

```
$ oc get managementbackup <bup-name> -n <APIC_namespace> -o yaml
```

Here is an example of some possible output:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementBackup
metadata:
  creationTimestamp: "2020-10-23T18:34:11Z"
  generation: 1
  labels:
    app.kubernetes.io/instance: m1
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: m1-e46bbac4
  name: m1-e46bbac4
  namespace: default
  resourceVersion: "6869"
  selfLink: /apis/management.apiconnect.ibm.com/v1beta1/namespaces/default/managementbackups/m1-e46bbac4
  uid: b7b6be9c-82d4-4737-984f-58584a3fe582
spec:
  clusterName: m1
  crType: create
  type: full
status:
  clusterName: m1-fbaa5be2-postgres
  conditions:
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupPending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    message: Record of management backup complete
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-23T18:34:11Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  crType: ""
  id: 20201023-183005F
  info:
    id: 20201023-183005F
    range: 2020-10-23 18:30:05 +0000 UTC - 2020-10-23 18:31:22 +0000 UTC
    rangeEnd: 1603477882
    rangeStart: 1603477805
    size: 32.6 MB
    totalsize: 32.6 MB
    type: full
  phase: Ready
  state: ""
  subsysName: m1
```

7. **Version 10.0.1.0 only**: If a local-backup was performed on v10.0.1.0, and you now wish to upload the local-backup file to an sftp server, see Uploading backup files to an sftp server for the required steps.

Restoring the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0

You can restore your Management subsystem in your OpenShift and Cloud Pak for Integration environments.

Before you begin

You must have backups of the management subsystem in order to restore. See Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0 and Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

About this task

To trigger a restore, you will create a **ManagementRestore** custom resource. When the **ibm-apiconnect** operator detects the new CR, it starts the restore process.

- Ensure that the backup method that you used is complete before you attempt a restore. The backup data that is stored in the remote server is used for restoring the Management subsystem data back to a previous state. Ensure that these backups are on your remote server before you attempt a restore, and that the backup ID in the selected backup exists in remote storage.
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.
- If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.

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

Procedure

1. Use the Form UI or the `oc` command to obtain a list of the available backups, and decide which one to restore your Management subsystem:

   ```
   oc get mgmtb -n <APIC_namespace>
   ```

   For example:

   ```
   $ oc get mgmtb
   NAME                   STATUS   ID                 CLUSTER      TYPE   CR TYPE   AGE
   management-3aa3bebf    Ready    20200929-152150F   management   full   record    10h
   management-3c4ca8df    Ready    20200929-115520F   management   full   record    20h
   management-5tbxj       Ready    20200929-224349F   management   full   create    17h
   management-f10af337    Ready    20200925-133703F   management   full   record    5d2h
   management-jtlcd       Ready    20200929-224349F   management   full   create    25h
   management-zxjtz       Ready    20200929-224349F   management   full   create    28h
   ```

   Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.
   You will use one of the values in the **NAME** column.

2. Use the instructions that apply to your release of API Connect:
   - Version 10.0.1.1 or later
     - Use the following code as an example:

       ```
       apiVersion: management.apiconnect.ibm.com/v1beta1
       kind: ManagementRestore
       metadata:
         generateName: management-
       spec:
         backupName: management-3aa3bebf
       ```

       Note that **backupName**: is the name of the backup custom resource you want to restore to. For example, using the output from the previous step:

       ```
       backupName: management-3aa3bebf
       ```

     - Use the Form UI or `oc` to create the ManagementRestore custom resource in the namespace of the Management Subsystem, to trigger the restore process:

       ```
       $ oc create -f mgmtrestore_cr.yaml -n <APIC_namespace>
       ```

     - You can list current restores using the following command, or if using OCP UI you can list all **ManagementResource** list:

       ```
       $ oc get managementrestore -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
       ```

       Example output:

       - SFTP:

         ```
         NAME                  STATUS      BACKUP               CLUSTER      MESSAGE
         PITR     AGE
         mgmt-restore-ptx6d    Complete    mgmt-backup-w4h8c    m1           Restore process completed (DB Restore +
         DRR job)             6m31s
         ```

       - S3

```
NAME                  STATUS      BACKUP               CLUSTER       MESSAGE
PITR                    AGE
mgmt-restore-hr9zx    Complete    mgmt-backup-xnjkl    m1            Restore process completed (DB Restore +
DRR job)    2020-11-23 23:08:32+00   9m8s
```

- Version 10.0.1.0
  - Use the following command to obtain the Point-in-Time-Recovery target for the selected backup:

    ```
    oc get mgmtb <backup_name> -o jsonpath="{..status.info.range}"
    ```

    For example:

    ```
    $ oc get mgmtb management-3aa3bebf -o jsonpath="{..status.info.range}"
    2020-09-29 15:21:50 +0000 UTC - 2020-09-29 15:30:56 +0000 UTC
    ```

    The target for recovery is the second half the **range** value in the backup custom resource. In the prior example: **2020-09-29 15:30:56 +0000**.

    The custom resource uses the **pitrTarget** setting to specify the Point-in-Time-Recovery target. You must set **pitrTarget** to be later than the value obtained above (such as by one second newer). Use the format **YYYY-MM-DD HH:MM:SS+Z**.

    Using the example above, where **2020-09-29 15:30:56 +0000** is the **range**, the **pitrTarget** is **2020-09-29 15:30:57+00**:

    ```
      pitrTarget: "2020-09-29 15:30:57+00"
    ```

  - Use the following code as an example:

    ```
    apiVersion: management.apiconnect.ibm.com/v1beta1
    kind: ManagementRestore
    metadata:
      generateName: management-
    spec:
      backupName: management-3aa3bebf
      pitrTarget: "2020-09-30 13:53:21+00" ## Not required in 10.0.1.1-eus and greater
    ```

    - **backupName** - The name of the backup custom resource you want to restore to. For example, using the output from the previous step, **management-3aa3bebf**.
    - **pitrTarget:** - The time you want to restore to, in format **YYYY-MM-DD HH:MM:SS+Z**.
  - Use the Form UI or **oc** to create the ManagementRestore custom resource in the namespace of the Management Subsystem, to trigger the restore process:

    ```
    $ oc create -f mgmtrestore_cr.yaml -n <APIC_namespace>
    ```

  - You can list current restores using the following command, or if using Form UI you can list all **ManagementResource** lists:

    ```
    $ oc get managementrestore -n <APIC_namespace> --sort-by=.metadata.creationTimestamp
    ```

    Example output:

    ```
    NAME                  STATUS      BACKUP               CLUSTER       PITR                       AGE
    management-6nzjr      Complete    mgmt-backup-dg4s4    m1            2020-01-05 19:05:00+01     14m
    ```

  - The operator creates an upgrade CR when the postgres database is restarted during the restore:

    ```
    managementdbupgrade.management.apiconnect.ibm.com/m1-up-zqkk5   Complete   Fresh install is Complete (DB
    Schema/data are up-to-date)
    63s5s
    ```

Troubleshooting management subsystem backups on OpenShift and Cloud Pak for Integration v10.0.6.0

You can troubleshoot failures of the API Connect management subsystem backups in an OpenShift or Cloud Pak for Integration environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- **Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Resolve resiliency issues with API Connect management system backups.
- **Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Troubleshoot configuration of s3 backups when the stanza-create job fails in an OpenShift or Cloud Pak for Integration environment.
- **Restore failure with compliance pod error on OpenShift and IBM Cloud Pak for Integration v10.0.6.0**
  Compliance pod reports an error, and restore does not progress.

Troubleshooting resiliency issues on OpenShift and Cloud Pak for Integration v10.0.6.0

Resolve resiliency issues with API Connect management system backups.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting **pgbackrest-shared-repo** pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to **local-storage**, then **pgbackrest-shared-repo** pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage, see: Monitoring Postgres disk usage on OpenShift or Monitoring Postgres disk usage on Cloud Pak for Integration.

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting the stanza-create job for S3 backup configuration on OpenShift and Cloud Pak for Integration v10.0.6.0

Troubleshoot configuration of s3 backups when the stanza-create job fails in an OpenShift or Cloud Pak for Integration environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
When you configure Management subsystem backups for s3 providers, the operator runs a **stanza-create** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Check the status of the **stanza-create** job:

```
oc get jobs -n <namespace> | grep stanza
```

On failure:

- **stanza-create** pods fail, and job status is **0/1**:

  ```
  oc get jobs | grep stanza
   m1-3bfe12ac-postgres-stanza-create          0/1           32m        32m
  ```

- Listing all pods brought up by **stanza-create** job shows multiple pods because a job tries **backoffLimit** times to complete the job. By default, there will be **7** pods in error state:

  ```
  oc get pods | grep stanza
   m1-3bfe12ac-postgres-stanza-create-726z4                      0/1     Error       0          11m
   m1-3bfe12ac-postgres-stanza-create-9vq4n                      0/1     Error       0          22m
   m1-3bfe12ac-postgres-stanza-create-gbtgb                      0/1     Error       0          30m
   m1-3bfe12ac-postgres-stanza-create-gl5g5                      0/1     Error       0          17m
   m1-3bfe12ac-postgres-stanza-create-q8qhw                      0/1     Error       0          26m
   m1-3bfe12ac-postgres-stanza-create-t84zs                      0/1     Error       0          8m54s
   m1-3bfe12ac-postgres-stanza-create-z8tfs                      0/1     Error       0          12m
  ```

  Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

  ```
  kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

  time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
  time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
  time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command
  requested"
  time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
  time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest
  stanza-create  --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --
  repo1-type=s3]"
  time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create  --
  db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3 "
  time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
  time="2021-04-13T20:05:21Z" level=info msg="output=[]"
  ```

- Job status is set to **type: Failed** and **status: "True"**, with **reason: BackoffLimitExceeded**:

  ```
  oc get job m1-3bfe12ac-postgres-stanza-create -o yaml

  status:
    conditions:
      - lastProbeTime: "2021-04-13T20:28:03Z"
        lastTransitionTime: "2021-04-13T20:28:03Z"
        message: Job has reached the specified backoff limit
        reason: BackoffLimitExceeded
        status: "True"
        type: Failed
        failed: 7
  startTime: "2021-04-13T20:01:18Z"
  ```

- If pod logs are lost, you can execute a command inside the **backrest-shared-repo** pod:

1. Obtain the `backrest-shared-repo` pod name:

   ```
   oc get pods -n <namespace> | grep backrest-shared-repo
   ```

   For example:

   ```
   oc get pods | grep backrest-shared-repo
   m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-twl95   1/1     Running
   ```

2. Exec into the pod:

   ```
   oc exec -it <backrest-shared-repo-pod> -- bash
   ```

3. Run:

   ```
   pgbackrest info --output json --repo1-type s3
   ```

   For example, for invalid hostname:

   ```
   ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
   ```

   The exact `ERROR:` will vary depending on the misconfiguration. Common errors:

   Invalid access key
   Invalid access key secret
   Invalid bucket region
   Invalid bucket or folder path

- To fix the incorrect settings, see Reconfiguring or adding backup settings after installation of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.

Restore failure with compliance pod error on OpenShift and IBM® Cloud Pak for Integration v10.0.6.0

Compliance pod reports an error, and restore does not progress.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. This failure can happen in a situation where a backup is taken on a system where API governance has never been enabled, and the restore of that backup is done onto a system which has API governance enabled. In this situation the API governance database is missing, and consequently the compliance pods report an error.

Symptoms

The compliance service pods are continually restarting, and the restore does not progress. The log of the compliance pod shows the following error:

```
2023-05-19T03:34:31.771Z bhendi:error Database availability check to management-0bfd2507-postgres:5432 failed, try again in 2000 ms,
error: : database "compliance" does not exist, stack: error: database "compliance" does not exist
```

Resolution

Delete all of the compliance jobs, and then start the restore again. For example:

```
oc -n <namespace> delete job management-up-compliance-service-data-populate-0-to-1 management-up-compliance-service-schema-0-to-1
```

Backing up and restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0

You can back up and restore your Developer Portal service in your OpenShift or Cloud Pak for Integration environment.

About this task

Before you can initiate a backup or restore operation, you must update the Portal CR (custom resource) to configure your settings. Generating a backup for Portal is performed on demand by deploying a backup CR. You can deploy a restore CR to start the restoration of a previously generated backup.

For points to consider when you are backing up and restoring a two data center disaster recovery deployment, see Backup and restore requirements for a 2DCDR deployment.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem.

- **Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Configure backups for the Developer Portal service in your OpenShift or Cloud Pak for Integration environment.
- **Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Perform a manual backup of the Developer Portal subsystem in your OpenShift or Cloud Pak for Integration environment.
- **Restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0**
  You can restore the Developer Portal subsystem and sites in your OpenShift or Cloud Pak for Integration environment.
- **Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Review details on a failed backup or restore operation for Developer Portal backup on OpenShift and Cloud Pak for Integration.

Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0

Configure backups for the Developer Portal service in your OpenShift or Cloud Pak for Integration environment.

About this task

Run commands in the namespace where the Developer Portal is installed.

For points to consider when you are backing up and restoring a two data center configuration, see Backup and restore requirements for a 2DCDR deployment.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Procedure

1. Create your Developer Portal backup secret.
   The backup secret is a Kubernetes secret that contains your credentials for accessing the s3 or sftp backup database.
   - S3
     Only password-based authentication is supported for S3, not authentication based on public certificates and private keys. Password-based authentication for S3 requires that you generate an access key and secret. For example:
     - IBM (Cloud Object Storage): Service credentials.
     - AWS: Managing access keys.

     The secret can be created with the following command:

     ```
     oc create secret generic portal-backup-secret --from-literal=username='Your_access_key-or-user_name'
         --from-literal=password='Your_access_key_secret-or-password' -n <Portal_namespace>
     ```

   - SFTP
     Supported credentials types:
     - User name and password (v10.0.2.0 or later).
     - User name and SSH-key (v10.0.3.0 or later). Only OpenSSH keys are supported.[1]

     Use one of the following commands to create the secret:
     - User name and password credentials

       ```
       oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>'
       --from-literal=password='<Your_password>' -n <Portal_namespace>
       ```

     - Version 10.0.3.0 or later: User name and SSH-key credentials:

       ```
       oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>'
        --from-file=ssh-privatekey='<Your_private_key_file>' -n <Portal_namespace>
       ```

2. Add a `portalBackup` subsection to the `spec.portal` section of the top-level `apiconnectcluster` CR.
   You can edit the YAML in the UI, or you can run the following command and then add the `portalBackup` block:

   ```
   oc edit apiconnectcluster <Portal_namespace>
   ```

   An example when you use `protocol: objstore`:

   ```
   spec:
     portal:
       portalBackup:
         credentials: portal-backup-secret
         host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
         path: test-bucket/restore-test
         port: 443
         backupCerts: <custom-s3-server-CA-cert>
         backups3URIStyle: <host-or-path>
         schedule: '0 2 * * *'
   ```

   An example when you use `protocol: sftp`:

   ```
   spec:
     portal:
   ```

```
  portalBackup:
    credentials:   portal-backup-secret
    host:          sftp-service.example.com
    path:          /home/fvtuser/site-backups
    port:          22
    protocol:      sftp
    schedule:      '0 2 * * *'
```

Notes about the **portalBackup:** entry:

- If **portalBackup:** entry is empty or omitted from the API Connect Cluster CR, then **local** backups will be taken if a backup CR is created. If the **schedule** exists, backups are taken at the scheduled time.
- For S3 and SFTP, the **portalBackup:** entry must contain host, credentials, protocol, and path.
- For S3, the **portalBackup:** section can optionally contain either, or both, of **backupCerts** and **backupS3URIStyle**.

Table 1. Developer Portal backup settings

Setting	Description

credentials	The name of the secret you created.
	Supported credential types:
	• S3
	Only password-based authentication is supported for S3, not authentication based on public certificates and private keys. Password-based authentication for S3 requires that you generate an access key and secret.
	• SFTP
	  • User name and password (v10.0.2.0 or later)
	  • User name and SSH-key (v10.0.3.0 or later)

host	• S3 - **host**: is the S3 endpoint with the corresponding S3 region in the format ***<S3endpoint>***, or with an optional S3 region in the format ***<S3endpoint>/<S3region>***. • SFTP - the hostname of the sftp server
path	• S3 - the name of your S3 bucket to store backup data, optionally followed by a **/** and then any subdirectories inside the bucket. Valid examples: **bucket** **bucket/path** **bucket/lots/of/paths** • SFTP - the full absolute path of the folder on the SFTP server, beginning with **/**. Attention: Ensure that your backup **path** has the correct permissions to allow your backup user to be able to read and write to it from within the Developer Portal.
port	The port for the protocol to connect to the host. • S3 - Ignored for S3 (because objstore always connects to port 443) • SFTP - Defaults to 22 if not specified • local - Ignored for local
protocol	The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values: • **sftp** - for secure file transfer protocol. • **objstore** - for S3 compatible object storage.

Setting	Description
`backupCerts`	(S3 storage only) **`backupCerts`** is the name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. Note: If you use a public certificate for the S3 storage provider, it must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: **`x509: certificate signed by unknown authority`** This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be **`ca.crt`** and the value is the base64-encoded value of the CA certificate. If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate. If you do need to provide the CA certificate then you must provide the entire chain in the **`ca.crt`** member of the secret, as follows: **`intermediate-certificate-1`** **`intermediate-certificate-2`** **`. . .`** **`intermediate-certificate-`*`n`*** **`root-certificate`** Where the first certificate in the **`ca.crt`** member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the **`ca.crt`** member is the issuer of the preceding certificate. The root certificate in the **`ca.crt`** member must be self-signed. If there are no intermediate certificates involved, then the **`ca.crt`** member contains only the root certificate. The following sample bash script creates the Kubernetes secret: **`cat >customs3ca.yaml <<EOF`** **`apiVersion: v1`** **`data:`** **`  ca.crt: $(base64 <`*`path-to-ca-certificate`*`> | tr -d '\n' )`** **`kind: Secret`** **`metadata:`** **`  name: custom-server-ca`** **`type: generic`** **`EOF`** **`kubectl apply -f customs3ca.yaml -n <`*`APIC_namespace`*`>`**
`backups3URIStyle`	(S3 storage only) Optional - **`backups3URIStyle`** indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, default to **`host`**. Supported beginning with 10.0.2. Valid values: **`host`** and **`path`**. Some custom S3 providers require URI style to be set to **`path`**. For example, MinIO supports both **`host`** and **`path`** style setup. You can create a MinIO S3 server to only accept **`path`** style client communications. Important: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.
`schedule`	The schedule for how often automatic Developer Portal backups are run. The format for the schedule is any valid cron string. The timezone for backups is UTC. Valid for all backup types: S3, SFTP, and local. **`spec:`** **`  portal:`** **`  portalBackup:`** **`  ...`** **`  schedule: '0 2 * * *'`**

- **Overview of the Developer Portal backup resources v10.0.6.0**
  How to manage your Developer Portal remote backup resources.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Overview of the Developer Portal backup resources v10.0.6.0

How to manage your Developer Portal remote backup resources.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Note: This information is relevant only if you're taking remote Developer Portal backups.
The Developer Portal uses two types of portal backup resources. When you run the command **`kubectl get portalbackup`**, you see a list of backups, for example:

```
NAME                   ID               TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE     COMMENT
my-backup-f91ms        n/a              20b47fe090f5 Ready    system   create    portal    78m
portal-record-222cg    20220108.010033               Ready    system   record    portal    94m
portal-record-22xdc    20220702.010339               Ready    site     record    portal    9m49s
portal-record-245j7    20221002.010340               Ready    site     record    portal    5m
portal-record-246hm    20221218.010103               Ready    site     record    portal    32m
portal-record-24h44    20220509.010037               Ready    site     record    portal    68m
portal-record-24wb4    20221003.010029               Ready    site     record    portal    60m
portal-record-264mh    20221119.010453               Ready    system   record    portal    78m
```

There are two types of resources listed, and these are denoted by the **CR TYPE** column; **create** or **record** types.

1. **create**: This resource is a representation of a **portalbackup** task that was created manually. See Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0, Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0, or Backing up the Developer Portal subsystem on VMware, for information about how to create a manual portal backup.
2. **record**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

   ```
   kubectl describe portalbackup portal-record-name
   ```

   For example:

   ```
   kubectl describe portalbackup portal-record-zmmfw
   .
   .
   Name:          portal-record-zmmfw
   Namespace:     apic
   Labels:        createdBy
   .
   .
   Full Backup Name:         my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
   ```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0

Perform a manual backup of the Developer Portal subsystem in your OpenShift or Cloud Pak for Integration environment.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Run commands in the namespace where the Developer Portal is installed.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem.
The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see Overview of the Developer Portal backup resources v10.0.6.0.

Procedure

1. Required: Backup the Management subsystem as explained in Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.
2. Create the **PortalBackup** CR (custom resource) as a YAML file with the backup options that you want to use for Developer Portal.
   You can use portalbackup_cr.yaml in the helper_files directory as an example:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalBackup
   metadata:
     generateName: portal-bup-
   spec:
     crType: create
     portalCluster: portal
     type: all
     comment: "test comment"
   ```

   Set the following backup properties as needed:

   - **crType** - Use **create** to trigger a backup process. When the backup is created, the operator generates another backup CR with **crType: record**, which represents the stored backup that you can use for a restore operation.
   - **portalCluster** - the name of the portal cluster you want to backup. You can find this with:

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

   - **record** - Includes a backup ID so you can choose it for restoring. If you plan to preserve the backup for use in recovering from a disaster, use the **record** argument to ensure that all necessary data is captured.
   - **type** - The extent of the backup; select one of the following options:
     - **all** - Backup the Developer Portal subsystem and all of the sites

    - **system** - Backup the Developer Portal subsystem
    - **site** - Backup only a specified Developer Portal site
  - **siteName**: Used only when you perform a **site** backup. Valid arguments are:
    - **site uuid** or **url** - back up only the specified site
    - **installed** - backup all installed sites
  - **comment**: Optional. You can optionally add a comment (such as a description or special notes) to the CR before you create a backup.
3. Run the following command to create the backup CR, which triggers the backup operation:

```
oc create -f portalbackup_cr.yaml -n <APIC_namespace>
```

4. Verify that the backup was successful by completing the following steps:
   a. Run the following command to a get a list of backups:

```
oc get portalbackup -n <APIC_namespace>
```

   The response looks like the following example, where only **crType: record** backups include an ID.

```
NAME               ID              STATUS     TYPE     CR TYPE   AGE     COMMENT
p1-bup-gqz8f    20200526.142449  Ready      system   record    55m
p1-bup-jx4vc    20200526.142458  Ready      site     record    54m
p1-bup-st4l2                     Ready      all      create    56m     y
p1-bup-th2dm                     Ready      all      create    4h51m   y
```

   b. Run the following command to view the details of a particular backup,, which is specified using the *<bup-name>* from the **NAME** column in the list of backups:

```
oc get portalbackup <bup-name> -n <APIC_namespace> -o yaml
```

   The details of a backup look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-
20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-
storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnnjnu
  message: ""
  phase: Ready
  state: ""
```

   c. Check the contents of the backup and make sure that it contains the following statements, which indicate a successful operation:

```
. . .
    reason: BackupComplete
    status: "True"
    type: Ready
```

If the operation did not complete successfully, see Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration v10.0.6.0 for information on reviewing logs to help resolve the problem.

Restoring Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0

You can restore the Developer Portal subsystem and sites in your OpenShift or Cloud Pak for Integration environment.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Run commands in the namespace where the Developer Portal is installed.

Important: Backups of the Management and Portal subsystems must be taken at the same time to ensure that the Portal sites are consistent with Management database. When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Developer Portal subsystem.

Procedure

1. Required: Restore up the Management subsystem as explained in Generating a manual backup of the management subsystem on OpenShift and Cloud Pak for Integration v10.0.6.0.
2. Choose which backup you want to restore.
   a. Run the following command to a get a list of available backups:

      ```
      oc get portalbackup -n <APIC_namespace>
      ```

      The response looks like the following example:

      ```
      NAME               ID             STATUS     TYPE     CR TYPE   AGE     COMMENT
      p1-bup-gqz8f    20200526.142449   Ready      system   record    55m
      p1-bup-jx4vc    20200526.142458   Ready      site     record    54m
      p1-bup-st4l2                      Ready      all      create    56m     y
      p1-bup-th2dm                      Ready      all      create    4h51m   y
      ```

      Note: You can only restore the Portal subsystem from backups where the **CR TYPE** is **record**.
   b. Run the following command to view the details of a particular backup,, which is specified using the *<bup-name>* from the **NAME** column in the list of backups:

      ```
      oc get portalbackup <bup-name> -n <APIC_namespace> -o yaml
      ```

      The details of a backup look like the following example:

      ```
      apiVersion: portal.apiconnect.ibm.com/v1beta1
      kind: PortalBackup
      metadata:
        creationTimestamp: "2020-10-01T10:00:16Z"
        generateName: portal-bup-
        generation: 1
        name: portal-bup-g66gv
        namespace: default
        resourceVersion: "8940"
        selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
        uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
      spec:
        comment: test comment
        portalCluster: portal
        crType: record
        siteName: installed
        type: site
      status:
        backupId: ""
        commentLeft: "y"
        conditions:
        - lastTransitionTime: "2020-10-01T10:00:18Z"
          reason: BackupFailed
          status: "False"
          type: Error
        - lastTransitionTime: "2020-10-01T10:01:19Z"
          message: |-
            2020-10-01 10:00:18: CLI task (backup:site) starting.
            2020-10-01 10:00:20: Making a backup for site conor.com
            2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-
      20201001.100021.tar.gz
            2020-10-01 10:00:30: Attempting to upload backup to remote backup server
            2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-
      storage.appdomain.cloud/eu-standard:22:cq-test/ptlcq4/ using objstore
            2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
            2020-10-01 10:00:33: All installed sites have been successfully backed up
            2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
          reason: BackupComplete
          status: "True"
          type: Ready
        - lastTransitionTime: "2020-10-01T10:01:19Z"
          reason: BackupInProgress
          status: "False"
          type: Running
        - lastTransitionTime: "2020-10-01T10:00:18Z"
          reason: BackupStatusUnknown
          status: "False"
          type: Warning
        fullBackupName: ""
        id: f8t0ux7z4zfnnjnu
        message: ""
      ```

```
  phase: Ready
  state: ""
```

3. Create the **PortalRestore** CR (custom resource) as a YAML file with the backup options that you want to use for Developer Portal. You can use portalrestore_cr.yaml in the helper_files directory as an example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  generateName: portal-restore-
spec:
  type: all
  portalCluster: portal
  dryRun: true
  timestamp: now
  priorityList:
    - myportal-critical-site.com
    - myportal-important.com
```

Add the following restore options as needed. Table 1 indicates which options are needed for each type of restore operation:

Table 1. The options that are needed for each type of Developer Portal restore operation

Option	Applies to `all`	Applies to `system`	Applies to `site`
`type`	X	X	X
`portalCluster`	X	X	X
`siteName`			X
`systemName`		X	
`timestamp`	X		
`dryRun`	X		
`priorityList`			

- **`type`**: Indicates the extent of the restore operation; specify one of the following options:
  - **`all`** - Restore the Developer Portal subsystem and all of the sites (supports only remote restores).
  - **`system`** - Restore the Developer Portal subsystem.
  - **`site`** - Restore only a specified Developer Portal site (you will indicate the site with the **`siteName`** option).
- **`portalCluster`** is the name of the portal cluster you want to restore. You can find this with:

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

- **`siteName`**: Indicates which site to restore (used only with the **`site`** option); specify one of the following options:
  - ***`ID`*** - Restore the backup specified by the ID. The ID must correspond to a Developer Portal backup of **`type: site`** and **`crType: record`**.
  - site-*1234*.tgz - Restore the site by using this file name (which must exist on the local file system or remote server).
  - myportal.com: find the latest backup for this URL (locally or remotely) and restore it.
  - **`all`** - Restore all sites that are found on the remote server by using the latest backup.
  - **`site`** - Restore only the site that matches the specified uuid or url.
  - **`installed`** - Restore all installed sites.
- **`systemName`** - Restore from a backup that corresponds to the specified system; use one of the following options:
  - ***`ID`*** - Restore the backup specified by the ID. The ID must correspond to a Developer Portal backup of **`type: site`** and **`crType: record`**.
  - system_backup-*1234*.tgz - Restore the system from the given file that exists on the pod's local file system or the remote server.
  - **`latest`** - Restore from latest subsystem backup that is found on either the local file system or the remote server.
- **`timestamp`** - Valid arguments are:
  - **`now`** - Restore using the latest backups available.
  - ***`< TIMESTAMP >`*** - Specified in ***`'YYYYMMDD.HHMMSS'`*** format to indicate a timestamp to retrieve the backup from. The nearest backup, searching backwards from this timestamp, is used.

  Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to ***`YYYYMMDD.HHMMSS`***. For Version 10.0.2.0 and earlier, the timestamp format is ***`YYYY-MM-DD HH:MM:SS`***.
- **`dryRun`**: Indicates whether an an actual restore operation is run; use one of the following options:
  - **`true`** - Execute a test, or dry run, of the command and return the list of actions that will be taken. These actions can be found in the custom resource.
  - **`false`** - Run a real restore operation, replacing the subsystem files and all sites. Any existing sites are reinstalled.
- **`priorityList`** - Restore sites in the sequence in which they are listed. The priority sites are queued for restoring first, followed by any remaining sites present in the backup location. Valid arguments are:
  - **`myportalsite.com`**
  - **`mysecondportalsite.com`**
- **`customPlatformApiHostname`** - If the **`API`** hostname of your management subsystem platform has changed since taking the backup you are restoring, then update it to the new platform API hostname of the management subsystem. (Optional)

4. Run the following command to create the restore CR, which triggers the restore operation:

```
oc create -f portalrestore_cr.yaml -n <APIC_namespace>
```

5. When the restore operation is finished, verify that the restore was successful cy completing the following steps:
   a. Run the following command to get the name of the most recent restore record:

   ```
   oc get portalrestore -n <APIC_namespace>
   ```

   The response looks like the following example:

   ```
   NAME                    READY     AGE
   portal-restore-twjrp    Ready     28s
   ```

   b. Run the following command to view information about a specific restore operation, which is indicated with the ***`<portal_restore_name>`*** shown in the **`NAME`** column:

```
oc get portalrestore <portal_restore_name> -n <APIC_namespace> -o yaml
```

The details of a restore look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-11-11T10:14:39Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-8f4c5
  namespace: portal
  resourceVersion: "1497597"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-8f4c5
  uid: 5a70b657-d998-4989-b9b6-8480e986311d
spec:
  siteName: "20200526.090209"
  portalCluster: portal
  type: site
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestorePending
    status: "False"
    type: Pending
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    message: |-
      2020-05-26 13:11:25: CLI task (restore:site) starting.
      2020-05-26 13:11:36: Deleting any old site data if it exists...
      2020-05-26 13:13:18: Restoring site test.com ...
      2020-05-26 13:13:18: .
      2020-05-26 13:13:19: Patching the database dump before restoring ...
      2020-05-26 13:13:48: .
      2020-05-26 13:19:41: Triggering the drush command to restore the site. This may take some time.
      2020-05-26 13:19:48: .
      2020-05-26 13:25:40: Drush restore command completed. Setting up the site.
      2020-05-26 13:25:48: .
      2020-05-26 13:30:49: Site test.com restored from backup /var/aegir/backups/test.com-20200526.090209.tar.gz.
Initializing the site...
      2020-05-26 13:30:52: Restore site completed.
      2020-05-26 13:30:52: CLI task (restore:site) completed successfully.
    reason: RestoreComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-11-11T14:55:32Z"
    reason: RestoreInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-11-11T10:14:39Z"
    reason: RestoreStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: bybyibd99wxtmkmk
  message: ""
```

c. Check the contents of the restore and make sure that it contains the following statements, which indicate a successful operation:

```
. . .
    reason: RestoreComplete
    status: "True"
    type: Ready
```

If the operation did not complete successfully, see Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration v10.0.6.0 for information on reviewing logs to help resolve the problem.

Troubleshooting Developer Portal backup and restore on OpenShift and Cloud Pak for Integration v10.0.6.0

Review details on a failed backup or restore operation for Developer Portal backup on OpenShift and Cloud Pak for Integration.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

Check the status of a backup operation

1. Run the following command to a get a list of backups:

   ```
   oc get portalbackup -n <Portal_namespace>
   ```

   The response looks like the following example, where only `crType: record` backups include an ID.

```
NAME                ID              STATUS     TYPE     CR TYPE   AGE     COMMENT
p1-bup-gqz8f    20200526.142449  Ready      system   record    55m
p1-bup-jx4vc    20200526.142458  Ready      site     record    54m
p1-bup-st4l2                     Ready      all      create    56m     y
p1-bup-th2dm                     Ready      all      create    4h51m   y
```

2. Run the following command to view the details of a particular backup,, which is specified using the *<bup-name>* from the **NAME** column in the list of backups:

```
oc get portalbackup <bup-name> -n <Portal_namespace> -o yaml
```

The details of a backup look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalBackup
metadata:
  creationTimestamp: "2020-10-01T10:00:16Z"
  generateName: portal-bup-
  generation: 1
  name: portal-bup-g66gv
  namespace: default
  resourceVersion: "8940"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/default/portalbackups/portal-bup-g66gv
  uid: b1d5b295-b966-48f2-b6ef-dfcd1c499741
spec:
  comment: test comment
  portalCluster: portal
  crType: create
  siteName: installed
  type: site
status:
  backupId: ""
  commentLeft: "y"
  conditions:
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupFailed
    status: "False"
    type: Error
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    message: |-
      2020-10-01 10:00:18: CLI task (backup:site) starting.
      2020-10-01 10:00:20: Making a backup for site conor.com
      2020-10-01 10:00:27: A local backup was succesfully created: /var/aegir/backups/conor.com-20201001.100021.tar.gz
      2020-10-01 10:00:30: Attempting to upload backup to remote backup server
      2020-10-01 10:00:30: Uploading conor.com-20201001.100021.tar.gz to s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-
standard:22:cq-test/ptlcq4/ using objstore
      2020-10-01 10:00:33: Site with URL conor.com (conor.com-20201001.100021.tar.gz) successfully uploaded
      2020-10-01 10:00:33: All installed sites have been successfully backed up
      2020-10-01 10:00:33: CLI task (backup:site) completed successfully.
    reason: BackupComplete
    status: "True"
    type: Ready
  - lastTransitionTime: "2020-10-01T10:01:19Z"
    reason: BackupInProgress
    status: "False"
    type: Running
  - lastTransitionTime: "2020-10-01T10:00:18Z"
    reason: BackupStatusUnknown
    status: "False"
    type: Warning
  fullBackupName: ""
  id: f8t0ux7z4zfnnjnu
  message: ""
  phase: Ready
  state: ""
```

3. If the operation failed, check the associated **message** for more information.

Check the status of a restore operation

1. Run the following command to get the name of the most recent restore record:

```
oc get portalrestore -n <Portal_namespace>
```

The response looks like the following example:

```
NAME                   READY     AGE
portal-restore-twjrp   Ready     28s
```

2. Run the following command to view information about the restore operation; which is indicated with the ***<portal_restore_name>*** shown in the **NAME** column:

```
oc get portalrestore <portal_restore_name> -n <Portal_namespace> -o yaml
```

The details of a failed restore operation look like the following example:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalRestore
metadata:
  creationTimestamp: "2020-05-26T13:10:46Z"
  generateName: portal-restore-
  generation: 1
  name: portal-restore-x2bf9
  namespace: portal
```

```
  resourceVersion: "1493471"
  selfLink: /apis/portal.apiconnect.ibm.com/v1beta1/namespaces/portal/portalrestores/portal-restore-x2bf9
  uid: 9f6a1e22-55ab-4595-8b98-bdfaec40d58c
spec:
  type: site
  portalCluster: portal
status:
  backupId: ""
  commentLeft: ""
  conditions:
  - lastTransitionTime: "2020-05-26T13:10:49Z"
    message: |
      Please provide a site to restore. To restore all sites use value 'all' in the CR
    status: Failed
    type: Ready
  fullBackupName: ""
  id: ""
  message: ""
```

3. If the operation failed, check the associated `message` for more information.

Review the logs for additional information

Look for information in the logs of the `ibm-apiconnect` pod:

- Locate the `ibm-apiconnect` pod with:

  `oc get pods -n <Portal_namespace>`

- View the logs with:

  `oc logs <ibm-apiconnect-pod-NAME> -n <Portal_namespace>`

Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration

The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

For information about analytics backup, restore, and disaster recover, refer to Backing up and restoring the analytics subsystem.

Disaster recovery V10.0.6.0

Prepare for, or recover from, a disaster affecting an API Connect deployment running on Kubernetes, OpenShift, or Cloud Pak for Integration.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. The backups that you create for disaster recovery are for recovery of the API Connect subsystems in the same environment that the backups were taken, or in a different environment that has the same network configuration and API Connect form factor. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see Migrating from v10 to v10 on a different form factor.

- **Disaster recovery on Kubernetes v10.0.6.0**
  Prepare for disaster events and recover API Connect on Kubernetes if a disaster event occurs.
- **Disaster recovery on OpenShift and Cloud Pak for Integration v10.0.6.0**
  Prepare for, or recover from, a disaster affecting an API Connect deployment running on OpenShift or Cloud Pak for Integration

Disaster recovery on Kubernetes v10.0.6.0

Prepare for disaster events and recover API Connect on Kubernetes if a disaster event occurs.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. You must complete preparation steps prior to a disaster in order to recover API Connect on Kubernetes.

Note: For information about how to maintain a two data center deployment, including failover and recovery procedures, see Maintaining a two data center deployment.

- **Preparing for a disaster v10.0.6.0**
  Prepare your API Connect deployment for disaster recovery in a Kubernetes environment.
- **Recovering from a disaster v10.0.6.0**
  Recover the API Connect subsystems on Kubernetes.

Preparing for a disaster v10.0.6.0

Prepare your API Connect deployment for disaster recovery in a Kubernetes environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

Begin preparing for disaster recovery by creating a back-up copy of the installation CR for each subsystem with the following commands:

1. `kubectl -n <namespace> get mgmt <mgmt-subsystem-name> -o yaml > mgmt-susbsystem-cr.yaml`
2. `kubectl -n <namespace> get a7s <a7s-subsystem-name> -o yaml > a7s-susbsystem-cr.yaml`
3. `kubectl -n <namespace> get ptl <ptl-subsystem-name> -o yaml > ptl-susbsystem-cr.yaml`

- **Preparing the management subsystem for disaster recovery v10.0.6.0**
  You can prepare a Management subsystem for disaster recovery from S3 or SFTP backups by taking specific steps before and after the installation of the subsystem.
- **Preparing the Developer Portal for disaster recovery v10.0.6.0**
  You can prepare a Developer Portal subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.
- **Preparing the Analytics subsystem for disaster recovery**
  Prepare the API Connect Analytics subsystem for disaster recovery in a Kubernetes environment.

Preparing the management subsystem for disaster recovery v10.0.6.0

You can prepare a Management subsystem for disaster recovery from S3 or SFTP backups by taking specific steps before and after the installation of the subsystem.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Management subsystem during the recovery process. Best practice is to complete these steps immediately after initial configuration of a management subsystem during your original v10 deployment.
- Any **local** backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Prior to initial installation, configure the Management subsystem for database backups:
   See Configuring backup settings for fresh install of the Management subsystem v10.0.6.0.
2. After installation, back up essential Kubernetes secrets that are used by the Management subsystem.
   The Kubernetes secrets are created during installation. You must save these in case a restore is needed after a disaster has occurred. At that time, you will use them in the setup of the restored Management subsystem:

   a. Obtain and save the Management database encryption secret.
      Use the following command to find the name of the secret in the status of the Management Subsystem. Replace **<namespace>** with the namespace used for the subsystem installation:

      ```
      kubectl get mgmt -n <namespace> -o yaml | grep encryption
      ```

      The output of this command shows the name of the Kubernetes secret storing the database encryption key, for example:

      ```
      encryptionSecret: management-enc-key
      ```

      In this case, the name of the Management database encryption secret is **management-enc-key**. Use the following command to back up this secret locally:

      ```
      kubectl get secret management-enc-key -n <namespace> -o yaml > management_enc_key.yaml
      ```

      The secret is stored in a file called **management_enc_key.yaml**, located in the present working directory.

   b. Obtain and save the Management client application credential secrets.
      i. Use the following command to find the names of the secrets in the status of the Management subsystem:

      ```
      kubectl get mgmt -n <namespace> -o yaml | grep CredentialSecret
      ```

      The output of this command shows the names of the Kubernetes Secrets used to store the various client application credentials. For example:

      ```
      atmCredentialSecret: management-atm-cred
      consumerToolkitCredentialSecret: management-ccli-cred
      consumerUICredentialSecret: management-cui-cred
      designerCredentialSecret: management-dsgr-cred
      juhuCredentialSecret: management-juhu-cred
      toolkitCredentialSecret: management-cli-cred
      uiCredentialSecret: management-ui-cred
      ```

      Here, for example, the name of the ATM Client Application Credential Secret is **management-atm-cred**.

      ii. Next, back up all of the secrets locally. Use the following command for each listed Credential Secret, replacing **<secret_name>** with the secret name listed each time:

      ```
      kubectl get secret <secret_name> -n <namespace> -o yaml > <secret_name>.yaml
      ```

      For example, use the following command to backup the ATM Client Credential Secret:

      ```
      kubectl get secret management-atm-cred -n <namespace> -o yaml > management-atm-cred.yaml
      ```

iii. After you save each of the client application Credential Secrets locally, open each of the YAML files saved for each secret in turn. Remove both the **ownerReferences** subsection and the **selfLink** property. Re-save the file.
For example, the **ownerReferences** and **selfLink** properties to be removed appear in the YAML files similar to the following:

```
ownerReferences:
  - apiVersion: management.apiconnect.ibm.com/v1beta1
    blockOwnerDeletion: true
    controller: true
    kind: ManagementCluster
    name: management
    uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
  selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
```

c. Ensure that all of the YAML files that contain the various backed-up Secrets are stored persistently and safely.
Important: If the files are lost, you cannot restore after a disaster event.
d. Take note of the following values in the Management subsystem CR. You will need them to restore the Management subsystem.

Table 1. Management subsystem CR settings

Setting	Used with Protocol	Example value
name	S3 and SFTP	**management**
siteName	S3 and SFTP	**82b290a2**
originalUID	S3 and SFTP	**fa0f6f49-b931-4472-b84d-0922a9a92dfd** : During restore, if you do not specify **originalUID**, the operator automatically sets its value in the new CR to the value of **metadata.uid**. If the **originalUID** in the new CR does not match **originalUID** in the saved CR, the restore will fail.
.databaseBackup.protocol	S3 and SFTP only	• S3: **objstore** • SFTP: **sftp**
databaseBackup.s3provider	S3 only	**aws**
databaseBackup.host	S3 and SFTP	• SFTP ***<SFTP-host-name>*** • S3: **s3.amazonaws.com/eu-west-1**
databaseBackup.path	S3 and SFTP	• SFTP ***<SFTP-path>*** • S3: **my-bucket/mgmt**
databaseBackup.schedule	S3 and SFTP	**0 3 * * ***
databaseBackup.credentials	SFTP only	***secret-containing-SFTP-credentials***
databaseBackup.backupCerts	S3 (custom only, v10.0.2.0 or later)	**my-custom-s3-ca-cert**
databaseBackup.backups3URIStyle	S3 (custom only, v10.0.2.0 or later)	**my-s3-uri-style**

- The values **name** and **siteName** form the name of the database cluster name, such as **management-82b290a2-postgres** and is needed to synchronize the backups from your old Management subsystem to your new one.
The **siteName** is either be a randomly generated alphanumeric ID, or a custom name that was specified by the administrator in the Management Subsystem CR helper YAML file at deploy time. Use the following command to get the **siteName**:

```
kubectl get mgmt -o yaml -n <namespace> | grep siteName
```

- The **databaseBackup.host** and **databaseBackup.path** settings must be identical to what was configured on your old Management subsystem. This is the location of your old Management subsystem backups from where we will recover from.
- Take note of **databaseBackupschedule**. The schedule will be included during the restoration.

3. Make sure that you have a backup that can be used in case of a disaster event:
   - If scheduled backups were configured as part of Configuring backup settings for fresh install of the Management subsystem v10.0.6.0, backups to the specified cloud storage will take place as scheduled.
   - You can also perform a manual backup at any time by creating a manual backup CR. See Generating a manual backup of the management subsystem v10.0.6.0.

What to do next

You should now complete the preparation steps for the Developer Portal subsystem; see Preparing the Developer Portal for disaster recovery v10.0.6.0.

Preparing the Developer Portal for disaster recovery v10.0.6.0

You can prepare a Developer Portal subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Developer Portal subsystem for the recovery process. Best practice is to complete these steps immediately after initial configuration of a Developer Portal subsystem during your original v10 deployment.
- Any **local** backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. For information about preparing the Management subsystem, see Preparing the management subsystem

for disaster recovery on Kubernetes. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Before the initial installation, configure the Developer Portal subsystem for database backups:
   a. Ensure that the Developer Portal backup subsystem secret is created in the cluster.
   This secret contains the credentials (username and password) for the remote storage, such as S3 or IBM Cloud Object Storage, that is used for storing the database backups.

   b. Ensure that the Developer Portal subsystem is configured for database backups.
   Add a **databaseBackups** subsection to the **spec** subsection of the Developer Portal subsystem Custom Resource (CR) YAML file. For example:

   ```
   portalBackup:
      credentials: portal-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: test-bucket/restore-test
      port: 443
      protocol: objstore
      schedule: ''
   ```

   See also: Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0
2. After installation, you must back up essential Kubernetes secrets that are used by the Developer Portal subsystem.
   The Kubernetes secrets are created during installation. You must save these secrets in case a restore is needed after a disaster. Then, you use them in the setup of the restored Developer Portal subsystem:

   a. Obtain and save the Developer Portal database encryption secret.
   Use the following command to find the name of the secret in the status of the Developer Portal Subsystem. Replace **<namespace>** with the namespace used for the subsystem installation:

   ```
   kubectl get portal -n <namespace> -o yaml | grep encryption
   ```

   The output of this command shows the name of the Kubernetes secret that is storing the database encryption key, for example:

   ```
   encryptionSecret: portal-enc-key
   ```

   In this case, the name of the Developer Portal database encryption secret is **portal-enc-key**. Use the following command to back up this secret locally:

   ```
   kubectl get secret portal-enc-key -n <namespace> -o yaml --export > portal_enc_key.yaml
   ```

   The secret is stored in a file that is called **portal_enc_key.yaml**, located in the present working directory.

   b. Obtain the endpoint definitions as they need to stay the same during the restore.
   Get the **portalAdminEndpoint** name:

   ```
   kubectl get portalcluster.portal.apiconnect.ibm.com portal -o jsonpath="{..portalAdminEndpoint.hosts[0].name}"
   ```

   Get the **portalUIEndpoint** name:

   ```
   kubectl get portalcluster.portal.apiconnect.ibm.com portal -o jsonpath="{..portalUIEndpoint.hosts[0].name}"
   ```

   Note: If you have more than one host specified in the **portalUIEndpoint** definition, you must get the name for each one.
   c. Ensure that all of these YAML files that contain the various backed-up Secrets are stored persistently and safely.
   Important: If the files are lost, you cannot restore after a disaster event.
3. Take note of the value of **originalUID** in the Portal subsystem CR. You will need it to restore the Portal subsystem. For example:

   ```
   spec:
     originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
   ```

   Note:
   - **Version 10.0.3.0 or later**: During restore, if you do not specify **originalUID**, the operator automatically sets its value in the new CR to the value of **metadata.uid**. If the **originalUID** in the new CR does not match **originalUID** in the saved CR, the restore will fail.
   - **Version 10.0.2 backups:** If planning to restore a Version 10.0.2 backup on Version 10.0.3.0 or later, make a note of the subsystem CR **metadata.uid**. You must use it when performing a restore on v10.0.3.0 or later, to populate the **spec.originalUID** value of the CR.
   - **Limitation for Version 10.0.2**: If planning to restore on Version 10.0.2, performing a restore may not work if the subsystem CR name exceeds 15 characters.
4. If Scheduled backups are set as part of the **databaseBackup** subsection in the Developer Portal Subsystem CR helper YAML file (**portal_cr.yaml**), backups to the specified cloud storage take place.
   You can also complete a manual backup at any time by creating a manual backup CR.

   a. To complete a manual backup, use the Developer Portal backup CR helper YAML file (**portalbackup_cr.yaml**). For example:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalBackup
   metadata:
     generateName: portal-bup-
   spec:
     type: all
     crType: create
     comment: "test comment"
   ```

   Run the following command to start a manual backup to the configured cloud storage:

   ```
   kubectl create -f portalbackup_cr.yaml -n <namespace>
   ```

   b. Use the following command to list the backups:

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

For example:

```
loginabc@localhost ibm-apiconnect]$ kubectl get pb
NAME                ID                STATUS   TYPE   CR TYPE   AGE     COMMENT
portal-bup-g66gv                      Ready    site   create    5m27s   y
portal-bup-z7wgk    20201001.100021   Ready    site   record    4m24s
```

With stored backups, secrets and **siteName** saved, the stack is now ready for a restore after a disaster event.

What to do next

To restore the Developer Portal on Kubernetes after a disaster, see Recovering the Developer Portal after a disaster v10.0.6.0.

Preparing the Analytics subsystem for disaster recovery

Prepare the API Connect Analytics subsystem for disaster recovery in a Kubernetes environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. For information about analytics backup, restore, and disaster recover, refer to Backup, restore, and disaster recovery.

Recovering from a disaster v10.0.6.0

Recover the API Connect subsystems on Kubernetes.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.

- **Recovering the management subsystem on Kubernetes v10.0.6.0**
  Recover the management subsystem on Kubernetes.
- **Recovering the Developer Portal after a disaster v10.0.6.0**
  You can restore the Developer Portal subsystem after a disaster event.
- **Recovering the Analytics subsystem after a disaster**
  Recover the API Connect analytics subsystem in a Kubernetes environment.

Recovering the management subsystem on Kubernetes v10.0.6.0

Recover the management subsystem on Kubernetes.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. The recovery procedures for the management subystem are specific to the type of backup. Use the instructions for your backup type:

- **Recovering the management subsystem from S3 backups v10.0.6.0**
  You can recover the management subsystem from S3 backups after a disaster event.
- **Recovering the management subsystem from SFTP backups v10.0.6.0**
  You can recover the management subsystem from SFTP backups after a disaster event.

Recovering the management subsystem from S3 backups v10.0.6.0

You can recover the management subsystem from S3 backups after a disaster event.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. To successfully recover the management subsystem, you must have previously completed the steps in Preparing the management subsystem for disaster recovery v10.0.6.0.
Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect® installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct **spec.originalUID** is specified upon restore. See Step 3.f.

Procedure

1. Determine which backup to restore from.
   - For IBM®'s Cloud Object Storage, you can check currently stored backups in the COS console. Select Buckets > Objects. See the Object Names displayed on the <*cos_name*>/backup/db panel. For example:

     ```
     20200605-105429F
     20200605-100008F
     20200605-11040F
     ```

   - For Amazon's AWS, you can check currently stored backups in the S3 console For example, under Amazon S3 > *cluster_name* > old cluster > backup > db:

     ```
     20200606-144315F
     20200606-145011F
     ```

   The format of the backup ID is **`YYYYMMDD-HHMMSS<F|I>`**

   Take note of the ID of the backup you wish to restore to. Each ID contains the date, time, and type of the backup stored. This will be used later in the procedure.

   a. Incremental backups are denoted with a suffix **`I`** on the ID, ensure each incremental backup has it's prior backup also present in storage. You can check this by examining the ID **`<prior-backup-id>_<backup-id>`**.
   b. Full backups are denoted with a suffix **`F`** on the ID.

   Note:
   During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same s3 bucket name in the database backup configurations.

2. Make sure you know the management database cluster name.
   You can get this name from the original management subsystem CR. You made note of this name in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

   If you are not able to recover the original management subsystem CR, use the following steps to recover the Management database cluster name:

   a. Open your IBM Cloud® Object Storage or AWS S3 console and proceed to the bucket location where the old Management subsystem backups are located.
   b. Download **`backup/db/<backup-id>/pg_data/postgresql.conf.gz`**. Open **`postgresql.conf`** to view the database cluster name:
      - IBM Cloud Object Storage

        ```
        .
        .
        cluster_name = 'm1-a6287572-postgres'
        .
        .
        ```

        - **`m1`** - Management subsystem name
        - **`a6287572`** - site name
      - AWS S3

        ```
        .
        .
        cluster_name = 'm1-c91dc0b9-postgres'
        .
        .
        ```

        - **`m1`** - Management subsystem name
        - **`c91dc0b9`** - site name

3. Before installing the replacement management subsystem CR:
   a. Apply the YAML file that contains the Management Database Encryption Secret into the cluster. For example, where **`encryption-bin-secret.yaml`** is the local YAML file containing the backup-up encryption secret:

      ```
      kubectl apply -f encryption-bin-secret.yaml -n <namespace>.
      ```

      Replace **`<namespace>`** with the namespace being used for the management subsystem installation.

      This command re-creates the original Management Database encryption secret on the cluster. It will be named as the original name of the secret.

   b. Add the following **`encryptionSecret`** subsection to the **`spec`** of the Management CR. For example, if **`management-enc-key`** is the name of the newly created secret on the cluster containing the original Management Database encryption secret from the previous step:

      ```
      encryptionSecret:
        secretName: management-enc-key
      ```

   c. For each of the saved YAML Files that contain the Management Client Application Credential Secrets, apply each file into the cluster using the following command:

      ```
      kubectl create -f <secret_name>.yaml -n <namespace>
      ```

      where **`<secret_name>`** is the local YAML file containing one of the backed-up Credential Secrets.

      Repeat this for each of the backed-up Credential Secrets. These are the secrets you saved n Step 2.b in Preparing the management subsystem for disaster recovery v10.0.6.0.

      These commands will re-create the original Management Client Application Credential Secrets on the cluster. Each will be named as the original name of the Secret.

   d. Add the following **`customApplicationCredentials`** subsection to the **`spec`** subsection of the Management CR:

      ```
      customApplicationCredentials:
      - name: atm-cred
        secretName: management-atm-cred
      ```

```
- name: ccli-cred
  secretName: management-ccli-cred
- name: cli-cred
  secretName: management-cli-cred
- name: cui-cred
  secretName: management-cui-cred
- name: dsgr-cred
  secretName: management-dsgr-cred
- name: juhu-cred
  secretName: management-juhu-cred
- name: ui-cred
  secretName: management-ui-cred
```

For each named credential above, the **secretName** is given as the corresponding name of the newly created secret from Step 3.c.

e. Add the **siteName** property to the **spec** of the Management CR.
For example, if **a2a5e6e2** is the original **siteName** that was noted after the installation of the original Management Subsystem:

```
siteName: a2a5e6e2
```

f. **Version 10.0.3.0 or later**: Add the **originalUID:** property to the **spec** of the Management CR.
When recreating a system, to restore a backup into it, you must specify the same **spec.originalUID** in the CR as was present in the system that was backed up. If the **spec.originalUID** in the new CR for recreating the system does not match the **spec.originalUID** that was present in the system that was backed up, the restore will fail.

```
spec:
  originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
```

Note:
- For Version 10.0.3.0 or later, if you do not specify **spec.originalUID** in the new CR, the operator automatically sets the Management CR value of **spec.originalUID** to match the new CR value **metadata.uid**. In this case, the restore will fail because the **spec.originalUID** in the saved (backed-up) CR does not match **spec.originalUID** in the new CR.
- The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Management.
- See also Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

g. Verify that the name of the management subsystem in the CR matches with the old management subsystem name, as described in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

4. Install the Management subsystem CR with the values obtained in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.
Important: The hostnames of the endpoints cannot be changed, and must remain the same in the Management CR YAML file used for installation now as they were for the original installation.
To review installation of the management subsystem, see Installing the Management subsystem.

   Once the Management subsystem is installed you may notice **backup** job pods and **stanza-create** job pods in **Error** state.

```
m1-82b290a2-postgres-stanza-create-4zcgz                  0/1     Error       0          35m
m1-82b290a2-postgres-full-sch-backup-2g9hm                0/1     Error       0          20m
```

   This is expected behavior.

   - The **stanza-create** job normally expects buckets or subdirectories within buckets to be empty. However, since we have configured the Management subsystem with a pre-populated bucket (ie. where our backups exist), the job will go into **Error** state.
   - Any scheduled or manual backups will go into **Error** state. While we have configured the Management subsystem with our already populated S3 bucket, the new database isn't yet configured to write backups into remote storage.

   Important:
   For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

   Note that if you delay completion of the restore:

   - Health check might fail. In this case, you can still proceed to the next step and perform a restore.
   - Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
     - Re-install the system, prepare again for disaster recovery, and perform the restore.
     - Or increase disk space so that the system returns to a stable state, and then proceed with the restore.

5. Get a list of available backups and confirm that the backup ID noted in Step 1 is in the backup list.
The API Connect Operator automatically reads backups from configured remote storage and populates the list of available backups we can restore to.

```
$ kubectl get mgmtb
NAME                STATUS     ID                 CLUSTER                  SUBSYSTEM   TYPE   CR TYPE   AGE
mgmt-backup-4z87f   Complete   20200606-145011F   m1-82b290a2-postgres   m1          full   record    11m
mgmt-backup-6bms2   Complete   20200606-144315F   m1-82b290a2-postgres   m1          full   record    11m
```

6. Update the Management subsystem CR **databaseBackup.schedule** to what was noted in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.
7. Perform a Management Restore using the name of the backup that has the ID you want to restore. For example, as shown in Step 5, for ID **20200606-145011F** the backup name is **mgmt-backup-4z87f**.
For more info on restoring the management subsystem, see Restoring the management subsystem v10.0.6.0.

8. Use the following command to check the status of the restore:

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

   Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Note:

   - Manual and scheduled backups should perform as normal once again
   - **stanza-create** jobs will continue to report **Error** state as stated in Step 4.

9. Verify that the Cloud Manager UI can now be logged-in to as before, and that Provider Orgs exist as before.

The restore from the disaster event is now complete.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on Kubernetes, see Recovering the Developer Portal after a disaster v10.0.6.0.

Recovering the management subsystem from SFTP backups v10.0.6.0

You can recover the management subsystem from SFTP backups after a disaster event.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
To successfully recover the management subsystem, you must have previously completed the steps in Preparing the management subsystem for disaster recovery v10.0.6.0.
Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct `spec.originalUID` is specified upon restore. See Step 3.f.

Procedure

1. Determine which backup to restore from.
   View the SFTP backups available on your remote storage site:

   ```
   -rw-r--r--    1 root     root     13092333 Aug 26 08:56 20200826-154646F.tgz
   -rw-r--r--    1 root     root     18703758 Aug 26 09:10 20200826-160010F.tgz
   -rw-r--r--    1 root     root     24318561 Aug 26 09:21 20200826-161301F.tgz
   ```

   Take note of the backup ID of the backup you wish to restore to. Each filename contains the date, time, and type of the backup stored. This will be used later in the procedure.

   The format of the backup ID is `YYYYMMDD-HHMMSS<F|I>`. For example, if we want to use the Aug 26, 09:21 backup, it's backup ID will be `20200826-161301F`
   a. Incremental backups are denoted with a suffix `I` on the ID, ensure each incremental backup has it's prior backup also present in storage. You can check this by examining the ID `<prior-backup-id>_<backup-id>`.
   b. Full backups are denoted with a suffix `F` on the ID.
2. Make sure you know the management database cluster name.
   You can get this name from the original management subsystem CR. You made note of this name in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

   If you are not able to recover the original management subsystem CR, you can also recover the Management database cluster name and siteName by examining the SFTP backup tar:

   a. Download or move the SFTP backup tar file and decompress (untar) it.
   b. Open `<management-subsystem-name>-<siteName>-postgres-backrest-shared-repo/backup/db/<backup-id>/pg_data/postgresql.conf.gz` which contains the management subsystem name and siteName. For example:

   ```
   # Do not edit this file manually!
   # It will be overwritten by Patroni!
   include 'postgresql.base.conf'

   archive_command = 'source /opt/cpm/bin/pgbackrest/pgbackrest-set-env.sh && pgbackrest archive-push "%p"'
   archive_mode = 'True'
   archive_timeout = '60'
   autovacuum_vacuum_cost_limit = '1000'
   autovacuum_vacuum_scale_factor = '0.01'
   cluster_name = 'm1-f785a3e3-postgres'
   ```

   In this example:

   - `cluster_name` has both the management subsystem name and siteName
   - `m1` - Management subsystem name
   - `f785a3e3` - site name
3. Before installing the replacement management subsystem CR:
   a. Apply the YAML file that contains the Management Database Encryption Secret into the cluster. For example, where `encryption-bin-secret.yaml` is the local YAML file containing the backup-up encryption secret:

   ```
   kubectl create -f encryption-bin-secret.yaml -n <namespace>.
   ```

Replace **<*namespace*>** with the namespace being used for the management subsystem installation.

This command re-creates the original Management Database encryption secret on the cluster. It will be named as the original name of the secret.

b. Add the following **encryptionSecret** subsection to the **spec** of the Management CR. For example, if **management-enc-key** is the name of the newly created secret on the cluster containing the original Management Database encryption secret from the previous step:

```
encryptionSecret:
  secretName: management-enc-key
```

c. For each of the saved YAML Files that contain the Management Client Application Credential Secrets, apply each file into the cluster using the following command:

```
kubectl create -f <secret_name>.yaml -n <namespace>
```

where **<secret_name>** is the local YAML file containing one of the backed-up Credential Secrets.

Repeat this for each of the backed-up Credential Secrets. These are the secrets you saved in Step 2.b in Preparing the management subsystem for disaster recovery v10.0.6.0.

These commands will re-create the original Management Client Application Credential Secrets on the cluster. Each will be named as the original name of the Secret.

d. Add the following **customApplicationCredentials** subsection to the **spec** subsection of the Management CR:

```
customApplicationCredentials:
- name: atm-cred
  secretName: management-atm-cred
- name: ccli-cred
  secretName: management-ccli-cred
- name: cli-cred
  secretName: management-cli-cred
- name: cui-cred
  secretName: management-cui-cred
- name: dsgr-cred
  secretName: management-dsgr-cred
- name: juhu-cred
  secretName: management-juhu-cred
- name: ui-cred
  secretName: management-ui-cred
```

For each named credential above, the **secretName** is given as the corresponding name of the newly created secret from Step 3.c.

e. Add the **siteName** property to the **spec** of the Management CR.
For example, if **a2a5e6e2** is the original **siteName** that was noted after the installation of the original Management Subsystem:

```
siteName: a2a5e6e2
```

f. **Version 10.0.3.0 or later**: Add the **originalUID:** property to the **spec** of the Management CR.
When recreating a system, to restore a backup into it, you must specify the same **spec.originalUID** in the CR as was present in the system that was backed up. If the **spec.originalUID** in the new CR for recreating the system does not match the **spec.originalUID** that was present in the system that was backed up, the restore will fail.

```
spec:
  originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
```

Note:

- For Version 10.0.3.0 or later, if you do not specify **spec.originalUID** in the new CR, the operator automatically sets the Management CR value of **spec.originalUID** to match the new CR value **metadata.uid**. In this case, the restore will fail because the **spec.originalUID** in the saved (backed-up) CR does not match **spec.originalUID** in the new CR.
- The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Management.
- See also Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

g. Verify that the name of the management subsystem in the CR matches with the old management subsystem name, as described in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0.

4. Install the Management subsystem CR with the values obtained in Step 2.d in Preparing the management subsystem for disaster recovery v10.0.6.0..
Important: The hostnames of the endpoints cannot be changed, and must remain the same in the Management CR YAML file used for installation now as they were for the original installation.
To review installation of the management subsystem, see Installing the Management subsystem.

5. Get a list of available backups and confirm that the backup ID noted in Step 1 is in the backup list.
The API Connect Operator automatically reads backups from configured remote storage and populates the list of available backups we can restore to.

```
$ kubectl get mgmtb
NAME                STATUS     ID                 CLUSTER                 SUBSYSTEM   TYPE   CR TYPE   AGE
mgmt-backup-4z87f   Complete   20200606-145011F   m1-82b290a2-postgres    m1          full   record    11m
mgmt-backup-6bms2   Complete   20200606-144315F   m1-82b290a2-postgres    m1          full   record    11m
```

6. Perform a Management Restore using the name of the backup that has the ID you want to restore. For example, as shown in Step 5, for ID **20200606-145011F** the backup name is **mgmt-backup-4z87f**.
For more info on restoring the management subsystem, see Restoring the management subsystem v10.0.6.0.

7. Use the following command to check the status of the restore:

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

Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Manual and scheduled backups should perform as normal once again.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on Kubernetes, see Recovering the Developer Portal after a disaster v10.0.6.0.

Recovering the Developer Portal after a disaster v10.0.6.0

You can restore the Developer Portal subsystem after a disaster event.

Before you begin

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

About this task

To recover from a disaster event, you must create a new IBM API Connect installation with a running IBM API Connect Operator.

Note: Limitation for backups created on Version 10.0.2:

- If restoring a Version 10.0.2 backup onto a new Version 10.0.2 deployment, performing a restore may not work if the subsystem CR name exceeds 15 characters. This limitation applies only to restoring onto Version 10.0.2.
- Restoration is supported for Version 10.0.2 backups onto a Version 10.0.3.0 or later deployment when subsystem name exceeds 15 characters, as long as the correct `spec.originalUID` is specified upon restore. See Step 1.f.

Procedure

1. To prepare for the restoration, you must complete some steps before you reinstall the Developer Portal subsystem.
   Warning:
   - Ensure that the hostnames of the endpoints remain the same in the Developer Portal subsystem that is used for installation now as they were for the original installation. The hostnames for the endpoints cannot be changed.

   a. Ensure that the Developer Portal backup subsystem secret is created in the cluster.
   This secret contains the credentials (username and password) for the remote storage, such as S3 or IBM Cloud Object Storage, that is used for storing the database backups. This secret must be the same secret as used for the original installation.

   b. Ensure that the Developer Portal subsystem is configured for database backups.
   Configuration is done by adding a **databaseBackups** subsection to the **spec** subsection of the Developer Portal subsystem Custom Resource (CR) YAML file, **portal_cr.yaml**. The file is located in the **helper_files** archive that was extracted as part of the preparation for installation. See Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0.

   For example:

   ```
   portalBackup:
         credentials: portal-backup-secret
         host: <backup-host>
         path: <backup-path>
         port: 22
         protocol: sftp
         schedule: ''
   ```

   The **host** and **path** must be the same as was specified for the original installation so that the IBM API Connect Operator can access the saved Developer Portal database backups.

   c. Apply the saved YAML file that contains the Developer Portal database encryption secret into the cluster.
   Use the following command, where **portal_enc_key.yaml** is the local YAML file that contains the backup encryption secret. Replace ***<namespace>*** with the namespace that is being used for subsystem installation:

   ```
   kubectl apply -f portal_enc_key.yaml -n <namespace>
   ```

   This command re-creates the original Developer Portal database encryption secret on the cluster, which is named as the original name of the secret.

   d. Add the **encryptionSecret** to the Developer Portal subsystem Custom Resource (CR) YAML file, **portal_cr.yaml**.
   For example, add the following **encryptionSecret** subsection to the **spec** section. In this example, **portal-enc-key** is the name of the new created secret on the cluster that contains the original Developer Portal Database Encryption Secret. The name of the secret that is created in Step 1.a is used here:

   ```
   encryptionSecret:
     secretName: portal-enc-key
   ```

   e. Ensure the name **spec.metadata.name** of the new Developer Portal cluster CR is the same as the previous Developer Portal CR.
   f. **Version 10.0.3.0 or later**: Add the **originalUID:** property to the **spec** of the Portal CR.
   When recreating a system, to restore a backup into it, you must specify the same **spec.originalUID** in the CR as was present in the system that was backed up. If the **spec.originalUID** in the new CR for recreating the system does not match the **spec.originalUID** that was present in the system that was backed up, the restore will fail.

   ```
   spec:
     originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
   ```

   Note:

- For Version 10.0.3.0 or later, if you do not specify `spec.originalUID` in the new CR, the operator automatically sets the Portal CR value of `spec.originalUID` to match the new CR value `metadata.uid`. In this case, the restore will fail because the `spec.originalUID` in the saved (backed-up) CR does not match `spec.originalUID` in the new CR.
- The originalUID is only essential when the subsystem CR name exceeds 15 characters in length, or 10 characters limit for the API Connect Cluster CR. Recommended practice is that all backups should include the originalUID for Portal.
- See also Preparing the Developer Portal for disaster recovery v10.0.6.0.

g. Ensure that the endpoint definitions are the same as the previous Developer Portal CR.

2. Apply the Developer Portal subsystem custom resource (CR) YAML file (`portal_cr.yaml`) to the cluster as documented as part of the original installation instructions.
   You can do this step now because the secrets are re-created, and additions made to `portal_cr.yaml`.

   Note: Wait for the Developer Portal subsystem to become fully up and running before proceeding.
3. Use the following command to check that the Developer Portal database backups that are stored in the configured cloud storage are now present as Developer Portal backup CRs:
   Note: You might wait for the cron job at the 10th minute `*/10 * * * *` to re-create your Developer Portal backup CRs.

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

   For example:

   ```
   loginabc@localhost ibm-apiconnect]$ kubectl get pb
   NAME              ID                STATUS   TYPE   CR TYPE   AGE     COMMENT
   portal-bup-g66gv                    Ready    site   create    5m27s   y
   portal-bup-z7wgk  20201001.100021   Ready    site   record    4m24s
   ```

   The `AGE` of each is new.

4. Perform a Developer Portal restore. See Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0
   For a disaster recovery scenario, it is advised that you restore all sites and systems. An example restore CR:

   ```
   apiVersion: portal.apiconnect.ibm.com/v1beta1
   kind: PortalRestore
   metadata:
     generateName: portal-restore-
   spec:
     type: all
     dryRun: false
     timestamp: now
     priorityList:
       - myportal-critical-site.com
       - myportal-important.com
   ```

5. When the Developer Portal restore CR is marked as `Ready`, the Disaster recovery is complete for the Developer Portal.

   ```
   kubectl get portalrestore
   ```

6. Verify that you can access the restored Developer Portal sites and that you are able to log in to each one.
   The restore from the disaster event is now complete.

Recovering the Analytics subsystem after a disaster

Recover the API Connect analytics subsystem in a Kubernetes environment.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. For information about analytics backup, restore, and disaster recover, refer to Backup, restore, and disaster recovery.

Disaster recovery on OpenShift and Cloud Pak for Integration v10.0.6.0

Prepare for, or recover from, a disaster affecting an API Connect deployment running on OpenShift or Cloud Pak for Integration

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. You must complete preparation steps prior to a disaster in order to recover API Connect.

- **Preparing for disaster recovery on OpenShift v10.0.6.0**
  Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.
- **Recovering from a disaster on OpenShift v10.0.6.0**
  Install and configure API Connect, then restore backed up data to recover the deployment after a disaster.
- **Preparing for disaster recovery on Cloud Pak for Integration v10.0.6.0**
  Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.
- **Recovering from a disaster on Cloud Pak for Integration v10.0.6.0**
  Install and configure API Connect, then restore the backed up data to recover the deployment after a disaster.

Preparing for disaster recovery on OpenShift v10.0.6.0

Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. To prepare your deployment for disaster, complete this task to copy the API Connect deployment settings, and then proceed to the following tasks to backup additional information for each subsystem.

Procedure

Complete the following steps to copy the API Connect deployment settings for use when you recover the deployment.

1. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

   ```
   oc get APIConnectCluster
   ```

   The response looks like the following example:

   ```
   NAME            READY   STATUS    VERSION    RECONCILED VERSION   AGE
   apic-cluster    7/7     Running   10.0.6   10.0.6-149         6d4h
   ```

2. Create a back-up copy of the installation CR for each subsystem, as well as the top-level APIConnectCluster CR, with the following commands:

   a. `oc -n <APIC_namespace> get apiconnectcluster apic-cluster-name -o yaml > apic-cluster-cr.yaml`

   b. `oc -n <APIC_namespace>> get mgmt apic-cluster-name-mgmt -o yaml > mgmt-susbsystem-cr.yaml`

   c. `oc -n <APIC_namespace> get a7s apic-cluster-name-a7s -o yaml > a7s-susbsystem-cr.yaml`

   d. `oc -n <APIC_namespace> get ptl apic-cluster-name-ptl -o yaml > ptl-susbsystem-cr.yaml`

What to do next

Proceed to Preparing the Management subsystem for disaster recovery on OpenShift v10.0.6.0.

1. Preparing the Management subsystem for disaster recovery on OpenShift v10.0.6.0
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
2. Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
3. Preparing the Analytics subsystem for disaster recovery on OpenShift
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Preparing the Management subsystem for disaster recovery on OpenShift v10.0.6.0

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save a copy of the Management database encryption secret.
   a. Run the following command to get the name of the encryption secret:

      ```
      oc get mgmt -n <APIC-namespace> -o yaml | grep encryption
      ```

      The output of this command shows the name of the secret that is storing the database encryption key, for example:

      ```
      encryptionSecret: management-enc-key
      ```

   b. Run the following command to save a local copy of the secret as a YAML file:

      ```
      oc get secret management-enc-key -n <APIC-namespace> -o yaml > management-enc-key.yaml
      ```

2. Save a copy of the Management client application credential secrets.
   Save the credential secrets (the **client_id** and **client_secret**) for client applications provided by the Management subsystem.

   a. Run the following command to get the names of the secrets:

      ```
      oc get mgmt -n <APIC-namespace> -o yaml | grep CredentialSecret
      ```

      The output of this command shows the names of the secrets used to store each of the client application credentials. For example:

      ```
      atmCredentialSecret: production-mgmt-atm-cred
      consumerToolkitCredentialSecret: production-mgmt-ccli-cred
      consumerUICredentialSecret: production-mgmt-cui-cred
      designerCredentialSecret: production-mgmt-dsgr-cred
      governanceCredentialSecret: production-mgmt-governance-cred
      juhuCredentialSecret: production-mgmt-juhu-cred
      toolkitCredentialSecret: production-mgmt-cli-cred
      uiCredentialSecret: production-mgmt-ui-cred
      ```

b. Run the following command to save a local copy of each secret as a YAML file, replacing **<secret_name>** with the name of the application credential secret:

```
oc get secret <secret_name> -n <APIC-namespace> -o yaml > <secret_name>.yaml
```

c. Edit each of the saved YAML files and remove both the **ownerReferences** subsection and the **selfLink** property, and then save the updated file.
The information to remove looks like the following example:

```
ownerReferences:
  - apiVersion: management.apiconnect.ibm.com/v1beta1
    blockOwnerDeletion: true
    controller: true
    kind: ManagementCluster
    name: management
    uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
  selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
```

3. Run a backup of the Management subsystem database.
   a. Configure the Management subsystem for backups.
   b. Perform a backup of the Management subsystem.
   c. Get the name and ID of the newest Management backup and save them for reference during a recovery.
   Run the following command to list the Management backups:

   ```
   oc get mgmtb -n <APIC-namespace>
   ```

   The response looks like the following example; the newest backup is the one with the smallest **AGE**:

   ```
   NAME                  STATUS   ID                 CLUSTER      TYPE   CR TYPE   AGE
   management-3aa3bebf   Ready    20200929-152150F   management   full   record    10h
   management-3c4ca8df   Ready    20200929-115520F   management   full   record    20h
   management-5tbxj      Ready    20200929-224349F   management   full   create    17h
   management-f10af337   Ready    20200925-133703F   management   full   record    5d2h
   management-jtlcd      Ready    20200929-224349F   management   full   create    25h
   management-zxjtz      Ready    20200929-224349F   management   full   create    28h
   ```

4. Save a copy of the username and password for a Cloud Manager administrator.
   When you recover the deployment, you will need to log in to the Cloud Manager as an administrator using the Cloud Manager User Registry. Any user with Administrator permission defined in the Cloud Manager User Registry can be used. Before backing up the information, verify that you an log in to Cloud Manager using that account.

   If you want to backup the secret that contains the administrator password that was created by the API Connect operator during the original deployment, run the following command to get the secret:

   ```
   oc --kubeconfig kubeconfig -n <APIC_namespace> get secret <instance-name>-mgmt-admin-pass -o jsonpath="{.data.password}" |
   base64 -d
   ```

   It is possible that the password was changed at some point and the secret is now out-of-date. Be sure to verify that you can log in to Cloud Manager with the password contained in the secret. If the password is not valid, then make sure you have another Cloud Manager administrator account in the Cloud Manager User Registry, and back up that username and password instead.

5. Copy all of the information that you saved to a secure location.

What to do next

Immediately proceed to Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0.

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.
Next topic: Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0

Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Perform a backup of the Management subsystem.

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save the ID of the Portal subsystem.
   If your original CR **name** was 10 characters or longer, you must add the **originalUID** value from the original Portal subsystem into the appropriate section of the installation CR during the recovery process.

When you deployed API Connect using the top-level APIConnectCluster CR, the deployment settings were added to subsystem-specific CRs that are managed by the top-level CR. When you recover the Portal subsystem after a disaster, you can locate the subsystem's ID in the saved copy of the Portal subsystem CR.

a. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

```
oc get APIConnectCluster
```

The response looks like the following example:

```
NAME           READY   STATUS    VERSION    RECONCILED VERSION   AGE
apic-cluster-name   7/7     Running   10.0.6   10.0.6-0         6d4h
```

b. Get the YAML that describes the Portal subsystem CR and save it in a file using the following command:

```
oc get ptl apic-cluster-name-ptl -o yaml >  apic-cluster-name-ptl.yaml
```

2. Back up the Developer Portal database encryption secret.
   a. Run the following command to get the name of the encryption secret:

```
oc get ptl -n <APIC-namespace> -o yaml | grep encryption
```

The output of this command shows the name of the secret that is storing the database encryption key, for example:

```
encryptionSecret: portal-enc-key
```

   b. Run the following command to save a local copy of the secret as a YAML file:

```
oc get secret portal-enc-key -n <APIC-namespace> -o yaml > portal-enc-key.yaml
```

3. Back up the Portal endpoint definitions.
   Save the information to a file because you must use the same names in the recovered deployment.

   a. Get the `portalAdminEndpoint` definition by running the following command:

```
oc get route | grep portal-director | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-director-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

   b. Get the `portalUIEndpoint` definition by running the following command:

```
oc get route | grep portal-web | awk '{print $2}'
```

The response looks like the following example:

```
dr-ptl-portal-web-apic.apps.vlxocp-1556.cp.fyre.ibm.com
```

4. Back up the Portal subsystem database.
   a. Configure the Portal subsystem for backups.
   b. Perform a backup of the Portal subsystem.
   Note: Make sure that you set the property `type: all` when you backup the Portal database, to ensure that you capture data for the Portal subsystem and all Portal sites.
   c. Get the name of the newest Portal backup and save it for reference during a recovery.
   Run the following command to list the Portal backups:

```
oc get portalbackup -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest **AGE**:

```
NAME                       ID    TASK ID             STATUS   TYPE   CR TYPE   CLUSTER          AGE   COMMENT
portal-manual-bup-9lghk    n/a   p5rkbbpyejrbqjab    Ready    all    create    production-ptl   15h   test comment
portal-manual-bup-jqbdm    n/a   cn8frkey1vgbls2u    Ready    all    create    production-ptl   15h   test comment
```

5. Copy all of the information that you saved to a secure location.

What to do next

Proceed to Preparing the Analytics subsystem for disaster recovery on OpenShift.

Previous topic: Preparing the Management subsystem for disaster recovery on OpenShift v10.0.6.0
Next topic: Preparing the Analytics subsystem for disaster recovery on OpenShift

Preparing the Analytics subsystem for disaster recovery on OpenShift

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. For information about analytics backup, restore, and disaster recover, refer to Backup, restore, and disaster recovery.

Previous topic: Preparing the Portal subsystem for disaster recovery on OpenShift v10.0.6.0

Recovering from a disaster on OpenShift v10.0.6.0

Install and configure API Connect, then restore backed up data to recover the deployment after a disaster.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. To successfully recover API Connect from a disaster, you must have previously completed the steps to prepare the Management, Portal, and Analytics subsystems for disaster recovery. Make sure that you can access the backups and deployment information because you cannot recover your deployment without it.

Attention: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal.

About this task

To recover from a disaster event, recreate your API Connect Cluster CR and add recovery details as explained in the following procedure. Use the file containing the original CR (which you saved during the preparation task) as a reference.

Important: Do not install API Connect until you have configured the CR as explained in this task.

Procedure

1. Make a local copy of the apiconnectcluster_cr.yaml template custom resource so that you can customize it for your deployment.
   Template files are stored in helper_files.zip. For both OpenShift and CP4I, you can configure settings directly in the YAML file as explained in this task. In CP4I, you will then copy the YAML code and paste it in using the YAML tab in IBM Cloud Pak Platform UI.

2. Configure backup and restore settings for each subsystem.
   Remember to create the backup secret for each subsystem as well, using the same name that you used in the original deployment (you can see the name in the **credentials** setting of each subsystem's backup section). The backup and restore settings, including the secrets, must exactly match the settings that you created in the original deployment. See the following topics for instructions on configuring backup and restore settings:

   - Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0
   - Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0
   - Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration

   Note:
   During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same s3 bucket name in the database backup configurations.

3. Apply the Management and Portal encryption secrets to the cluster.
   a. Apply the YAML file that contains the Management database encryption secret into the cluster.
      Run the following command, using the filename of the encryption secret that you saved when you prepared the Management subsystem for disaster recovery:

      ```
      oc -n <APIC_namespace> apply -f mgmt-enc-key.yaml
      ```

   b. Apply the YAML file that contains the Portal encryption secret into the cluster.
      Run the following command, using the filename of the encryption secret that you saved when you prepared the Portal subsystem for disaster recovery:

      ```
      oc -n <APIC_namespace> apply -f portal-enc-key.yaml
      ```

4. Important: Make sure that your new version of the API Connect Cluster installation CR uses the same value for the **name** setting as the original CR.
   Edit the YAML file that you copied from the template in the helper_files.zip. In the following steps, you will configure additional deployment and recovery settings in the new API Connect Cluster installation CR before you install API Connect.

5. Add the Management and Portal encryption secrets to the API Connect Cluster CR.
   Add each secret to the appropriate **spec.*subsystem*** section of the CR, as shown in the following example:

   ```
   spec:
     management:
       encryptionSecret:
         secretName: mgmt-enc-key
     portal:
       encryptionSecret:
         secretName: portal-enc-key
   ```

6. Apply each of the Management client application credential secrets to the cluster.
   Run the following command to apply each secret, using the filename of each client application credential secret that you saved when you prepared the Management subsystem for disaster recovery:

   ```
   oc -n <APIC_namespace> apply -f <secretName>.yaml
   ```

   For example, a deployment might use the following names for the secrets:

   ```
   atmCredentialSecret: management-atm-cred
   consumerToolkitCredentialSecret: management-ccli-cred
   consumerUICredentialSecret: management-cui-cred
   designerCredentialSecret: management-dsgr-cred
   governanceCredentialSecret: management-governance-cred
   juhuCredentialSecret: management-juhu-cred
   toolkitCredentialSecret: management-cli-cred
   uiCredentialSecret: management-ui-cred
   ```

7. Add the credential secrets to the **spec.management** section of API Connect Cluster CR.
   For example:

```
spec:
  management:
    customApplicationCredentials:
    - name: atm-cred
      secretName: management-atm-cred
    - name: governance-cred
      secretName: management-governance-cred
    - name: ccli-cred
      secretName: management-ccli-cred
    - name: cui-cred
      secretName: management-cui-cred
    - name: dsgr-cred
      secretName: management-dsgr-cred
    - name: juhu-cred
      secretName: management-juhu-cred
    - name: cli-cred
      secretName: management-cli-cred
    - name: ui-cred
      secretName: management-ui-cred
```

8. Add the **siteName** property to the **spec.management** and **spec.portal** sections of the API Connect Cluster CR.
   For example, if the **a2a5e6e2** is the original **siteName** of the Management subsystem, and **890772e3** is the original **siteName** of the Developer Portal subsystem, the CR looks like the following example:

   ```
   spec:
     management:
       siteName: "a2a5e6e2"
     portal:
       siteName: "890772e3"
   ```

9. If your original CR **name** was 10 characters or longer, you must add the **originalUID** value from the original Management and Portal subsystems into the appropriate sections of the installation CR.
   Important: If your original CR name was 10 characters or longer and you do not add this value, the deployments and routes that are generated during installation will include a different UID, which will cause the recovery process to fail.
   The IDs were created with the original deployment of the Management and Portal subsystems, and the recovered subsystems must continue to use the same IDs. You can locate the ID in the Management (*apic-cluster-name*-mgt.yaml) or Portal (*apic-cluster-name*-ptl.yaml) CR that you saved while preparing for disaster recovery.

   Locate the original setting in the appropriate subsystem CR, and copy them to corresponding section of the new API Connect Cluster CR using the same key and value.

   For example:

   ```
   spec:
     management:
       originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
     portal:
       originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
   ```

10. If you installed the API Connect using an S3 provider for the Management subsystem backups, add the following annotation to the **apiconnectcluster** CR.
    If you specified SFTP or local as the backup, skip this step.

    - Using the Platform UI:
      - Edit the API Management instance and click the YAML tab to edit the CR.
      - Add the following statement to the **spec.metadata.annotations** section:

        ```
        apiconnect-operator/deployment-mode: disasterRecovery
        ```

        For example:

        ```
        metadata:
          annotations:
            apiconnect-operator/deployment-mode: disasterRecovery
        ```

    - Using the CLI:
      - Get the name of the CR by running the following command:

        ```
        oc get apiconnectcluster -o name -n <APIC_namespace>
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name
        ```

      - Add the following annotation to the **spec.metadata.annotations** section of the CR by running the following command:

        ```
        oc annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode="disasterRecovery" -n
        <APIC_namespace>
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name annotated
        ```

11. Verify that you correctly added all of the settings to the API Connect Cluster CR.
    For example, the completed CR might look like the following example:

    ```
    apiVersion: apiconnect.ibm.com/v1beta1
    kind: APIConnectCluster
    metadata:
      namespace: apiconnect
      name: apis-minimum
      labels:
        app.kubernetes.io/instance: apiconnect
    ```

```
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
  annotations:
    apiconnect-operator/deployment-mode: disasterRecovery
spec:
  license:
    accept: true
    license: L-RJON-C2YLGB
    metric: VIRTUAL_PROCESSOR_CORE
    use: nonproduction
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
    encryptionSecret:
      secretName: apis-minim-faaa44bf-enc-key
    siteName: "faaa44bf"
    customApplicationCredentials:
      - name: atm-cred
      - secretName: apis-minim-faaa44bf-atm-cred
      - name: governance-cred
      - secretName: management-governance-cred
      - name: ccli-cred
      - secretName: apis-minim-faaa44bf-ccli-cred
      - name: cli-cred
      - secretName: apis-minim-faaa44bf-cli-cred
      - name: cui-cred
      - secretName: apis-minim-faaa44bf-cui-cred
      - name: dsgr-cred
      - secretName: apis-minim-faaa44bf-dsgr-cred
      - name: juhu-cred
      - secretName: apis-minim-faaa44bf-juhu-cred
      - name: ui-cred
      - secretName: apis-minim-faaa44bf-ui-cred
    analytics:
      client: {}
      ingestion: {}
    apiManagerEndpoint: {}
    cloudManagerEndpoint: {}
    consumerAPIEndpoint: {}
    platformAPIEndpoint: {}
    portal:
      admin: {}
    databaseBackup:
      path: ocp-dr-mgmt
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      s3provider: ibm
      schedule: 0 1 * * *
      protocol: objstore
      credentials: mgmt-backup-secret
  analytics:
    storage:
      enabled: true
      type: unique
    databaseBackup:
      chunkSize: 1GB
      credentials: a7s-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: ocp-dr-a7s
      schedule: 0 1 * * *
  storageClassName: rook-ceph-block
  profile: n1xc7.m48
  portal:
    adminClientSubjectDN: ""
    portalAdminEndpoint: {}
    portalUIEndpoint: {}
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
    encryptionSecret:
      secretName: apis-minim-913edd20-enc-key
    siteName: "913edd20"
    portalBackup:
      credentials: portal-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: ocp-dr-portal
      protocol: objstore
      schedule: 0 1 * * *
  version: 10.0.3
```

12. Prepare your environment for installing API Connect by completing the prerequisites and all components of step 1 in Installing API Connect.
    After you prepare your environment, continue with this procedure to install API Connect and then restore data.

13. Install API Connect by creating the CR with the following command:
    Replace ***<cr_name>*** with the value of the **name** setting in the API Connect Cluster CR.

    ```
    oc -n <APIC_namespace> create -f <cr_name>.yaml
    ```

14. Wait for **all** subsystems to be created.
    When the Management subsystem is installed, you might see **backup** job pods and **stanza-create** job pods in **Error** state; for example:

    ```
    m1-82b290a2-postgres-stanza-create-4zcgz                  0/1     Error       0          35m
    m1-82b290a2-postgres-full-sch-backup-2g9hm                0/1     Error       0          20m
    ```

    This is expected behavior, for the following reasons:

    - The **stanza-create** job normally expects buckets or subdirectories within buckets to be empty. However, since you configured the Management subsystem with your already-populated S3 bucket (where your backups exist), the job will enter the **Error** state.

- Any scheduled or manual backups will enter the `Error` state. Although you configured the Management subsystem with your already-populated S3 bucket, the new database isn't yet configured to write backups into remote storage.

The errors will not prevent a successful restore, so continue to the next step.

Important:
For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

Note that if you delay completion of the restore:

- Health check might fail. In this case, you can still proceed to the next step and perform a restore.
- Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
  - Re-install the system, prepare again for disaster recovery, and perform the restore.
  - Or increase disk space so that the system returns to a stable state, and then proceed with the restore.

15. Restore the subsystems in the following sequence:
    a. Restore the Management subsystem.
    When you perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Portal subsystem.

    Important: Be sure to restore the Management subsystem from the backup whose name and ID of the you noted while preparing for disaster recovery.
    b. Restore the Developer Portal subsystem.
    Use the restore type `all` to ensure that you restore the complete subsystem and all Portal sites.

    c. Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration.
16. Verify that the recovery was successful:
    a. Ensure that you can log in to the Cloud Manager UI.
    b. Verify that your provider organizations exist.
    c. Ensure that you can log in to each Developer portal.
    d. Ensure that the Analytics dashboard contains all of the analytics data that you preserved.
17. After the successful recovery, remove the annotation that you added to the `apiconnectcluster` CR in step 10.
    If you skipped step 10, then skip this step as well.

    - Using the Platform UI:
      - Edit the API Management instance and click the YAML tab to edit the CR.
      - Delete the following statement from the `spec.metadata.annotations` section:

        ```
        apiconnect-operator/deployment-mode: disasterRecovery
        ```

    - Using the CLI:
      - Get the name of the CR by running the following command:

        ```
        oc get apiconnectcluster -o name -n <APIC_namespace>
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name
        ```

      - Remove the annotation by running the following command, making sure to include the trailing "-" on `deployment-mode-` to indicate the removal:

        ```
        oc -n <APIC_namespace> annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode-
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name annotated
        ```

Preparing for disaster recovery on Cloud Pak for Integration v10.0.6.0

Back up data and essential deployment information from each API Connect subsystem so that you can use it to recover your deployment after a disaster.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
To prepare your deployment for disaster, complete this task to copy the API Connect deployment settings, and then proceed to the following tasks to backup additional information for each subsystem.

Procedure

Complete the following steps to copy the API Connect deployment settings for use when you recover the deployment.

1. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

   ```
   oc get APIConnectCluster
   ```

   The response looks like the following example:

   ```
   NAME           READY   STATUS    VERSION    RECONCILED VERSION   AGE
   apic-cluster   7/7     Running   10.0.6   10.0.6-149         6d4h
   ```

2. Create a back-up copy of the installation CR for each subsystem, as well as the top-level APIConnectCluster CR, with the following commands:

   a. `oc -n <APIC_namespace> get apiconnectcluster apic-cluster-name -o yaml > apic-cluster-cr.yaml`

   b. `oc -n <APIC_namespace>> get mgmt apic-cluster-name-mgmt -o yaml > mgmt-susbsystem-cr.yaml`

c. `oc -n <APIC_namespace> get a7s apic-cluster-name-a7s -o yaml > a7s-susbsystem-cr.yaml`

d. `oc -n <APIC_namespace> get ptl apic-cluster-name-ptl -o yaml > ptl-susbsystem-cr.yaml`

What to do next

Proceed to Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0.

1. Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
2. Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.
3. Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0
   Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

About this task

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery. Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save a copy of the Management database encryption secret.
   a. Run the following command to get the name of the encryption secret:

      ```
      oc get mgmt -n <APIC-namespace> -o yaml | grep encryption
      ```

      The output of this command shows the name of the secret that is storing the database encryption key; for example:

      ```
      encryptionSecret: management-enc-key
      ```

   b. Run the following command to save a local copy of the secret as a YAML file:

      ```
      oc get secret management-enc-key -n <APIC-namespace> -o yaml > management-enc-key.yaml
      ```

2. Save a copy of the Management client application credential secrets.
   Save the credential secrets (the **client_id** and **client_secret**) for client applications provided by the Management subsystem.

   a. Run the following command to get the names of the secrets:

      ```
      oc get mgmt -n <APIC-namespace> -o yaml | grep CredentialSecret
      ```

      The output of this command shows the names of the secrets used to store each of the client application credentials; for example:

      ```
      atmCredentialSecret: management-atm-cred
      consumerToolkitCredentialSecret: management-ccli-cred
      consumerUICredentialSecret: management-cui-cred
      designerCredentialSecret: management-dsgr-cred
      governanceCredentialSecret: management-governance-cred
      juhuCredentialSecret: management-juhu-cred
      toolkitCredentialSecret: management-cli-cred
      uiCredentialSecret: management-ui-cred
      ```

   b. Run the following command to save a local copy of each secret as a YAML file, replacing **<secret_name>** with the name of the application credential secret:

      ```
      oc get secret <secret_name> -n <APIC-namespace> -o yaml > <secret_name>.yaml
      ```

   c. Edit each of the saved YAML files and remove both the **ownerReferences** subsection and the **selfLink** property, and then save the updated file.
      The information to remove looks like the following example:

      ```
      ownerReferences:
        - apiVersion: management.apiconnect.ibm.com/v1beta1
          blockOwnerDeletion: true
          controller: true
          kind: ManagementCluster
          name: management
          uid: 623e6b20-7eb8-46ce-94ac-6b64cd71afc4
        selfLink: /api/v1/namespaces/default/secrets/management-atm-cred
      ```

3. Save a copy of the CP4I credentials secret.
   a. Run the following command to get the name of the encryption secret and save a local copy of the secret as a YAML file:

      ```
      oc get secrets <instance_name>-cp4i-creds -o yaml > <cp4i_creds_secret>.yaml
      ```

   b. Edit the saved YAML file and remove both the **ownerReferences** subsection and the **selfLink** property, and then save the updated file.
4. Run a backup of the Management subsystem database.
   a. Configure the Management subsystem for backups.

b. Perform a backup of the Management subsystem.
c. Get the name and ID of the newest Management backup and save them for reference during a recovery.
Run the following command to list the Management backups:

```
oc get mgmtb -n <APIC-namespace>
```

The response looks like the following example; the newest backup is the one with the smallest **AGE**:

```
NAME                   STATUS   ID                 CLUSTER      TYPE   CR TYPE   AGE
management-3aa3bebf    Ready    20200929-152150F   management   full   record    10h
management-3c4ca8df    Ready    20200929-115520F   management   full   record    20h
management-5tbxj       Ready    20200929-224349F   management   full   create    17h
management-f10af337    Ready    20200925-133703F   management   full   record    5d2h
management-jtlcd       Ready    20200929-224349F   management   full   create    25h
management-zxjtz       Ready    20200929-224349F   management   full   create    28h
```

5. Save a copy of the username and password for a Cloud Manager administrator.
When you recover the deployment, you will need to log in to the Cloud Manager as an administrator using the Cloud Manager User Registry. Any user with Administrator permission defined in the Cloud Manager User Registry can be used. Before backing up the information, verify that you can log in to Cloud Manager using that account.

   If you want to backup the secret that contains the administrator password that was created by the API Connect operator during the original deployment, run the following command to get the secret:

```
oc -n <APIC_namespace> get secret <instanceName>-mgmt-admin-pass -o jsonpath="{.data.password}" | base64 -d
```

   It is possible that the password was changed at some point and the secret is now out-of-date. Be sure to verify that you can log in to Cloud Manager with the password contained in the secret. If the password is not valid, then make sure you have another Cloud Manager administrator account in the Cloud Manager User Registry, and back up that username and password instead.

6. Copy all of the information that you saved to a secure location.

What to do next

Immediately proceed to Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0.

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.
Next topic: Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Perform a backup of the Management subsystem.

Important: Backups of the Management and Portal subsystems must be performed in sequence, and close together in time, to ensure that the Portal sites are consistent with Management database.

About this task

Make sure to store all backups and deployment information to a secure location where you can access it after a disaster. You cannot recover a deployment without the information that you collect during this task.

Procedure

1. Save the ID of the Portal subsystem.
If your original CR **name** was 10 characters or longer, you must add the **originalUID** value from the original Portal subsystem into the appropriate section of the installation CR during the recovery process.

   When you deployed API Connect using the top-level APIConnectCluster CR, the deployment settings were added to subsystem-specific CRs that are managed by the top-level CR. When you recover the Portal subsystem after a disaster, you can locate the subsystem's ID in the saved copy of the Portal subsystem CR.

   a. Find the API Connect cluster's name by running the following command and looking for the value in the NAME column:

```
oc get APIConnectCluster
```

   The response looks like the following example:

```
NAME            READY   STATUS    VERSION    RECONCILED VERSION   AGE
apic-cluster-name   7/7     Running   10.0.6   10.0.6-149          6d4h
```

   b. Get the YAML that describes the Portal subsystem CR and save it in a file using the following command:

```
oc get ptl apic-cluster-name-ptl -o yaml >  apic-cluster-name-ptl.yaml
```

2. Back up the Developer Portal database encryption secret.
   a. Run the following command to get the name of the encryption secret:

      ```
      oc get ptl -n <APIC-namespace> -o yaml | grep encryption
      ```

      The output of this command shows the name of the secret that is storing the database encryption key, for example:

      ```
      encryptionSecret: portal-enc-key
      ```

   b. Run the following command to save a local copy of the secret as a YAML file:

      ```
      oc get secret portal-enc-key -n <APIC-namespace> -o yaml > portal-enc-key.yaml
      ```

3. Back up the Portal endpoint definitions.
   Save the information to a file because you must use the same names in the recovered deployment.

   a. Get the `portalAdminEndpoint` definition by running the following command:

      ```
      oc get route | grep portal-director | awk '{print $2}'
      ```

      The response looks like the following example:

      ```
      dr-ptl-portal-director-apic.apps.vlxocp-1556.cp.fyre.ibm.com
      ```

   b. Get the `portalUIEndpoint` definition by running the following command:

      ```
      oc get route | grep portal-web | awk '{print $2}'
      ```

      The response looks like the following example:

      ```
      dr-ptl-portal-web-apic.apps.vlxocp-1556.cp.fyre.ibm.com
      ```

4. Back up the Portal subsystem database.
   a. Configure the Portal subsystem for backups.
   b. Perform a backup of the Portal subsystem.
      Note: Be sure to `type: all` when you backup the Portal database, to ensure that you capture data for the Portal subsystem and all Portal sites.
   c. Get the ID of the latest `all` Portal backup and save it for reference during a recovery.
      Run the following command to list the Portal backups:

      ```
      oc get portalbackup -n <APIC-namespace>
      ```

      The response looks like the following example; the latest backup is the one with a `TYPE` of `all` and the smallest `AGE`:

      ```
      NAME                ID              STATUS      TYPE      CR TYPE    AGE      COMMENT
      p1-bup-gqz8f    20200526.142449  Ready       system    record     55m
      p1-bup-jx4vc    20200526.142458  Ready       site      record     54m
      p1-bup-st4l2                     Ready       all       create     56m      y
      p1-bup-th2dm                     Ready       all       create     4h51m    y
      ```

5. Copy all of the information that you saved to a secure location.

What to do next

Proceed to Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0.

Previous topic: Preparing the Management subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0
Next topic: Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Preparing the Analytics subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Back up data and essential deployment information so that you can use it to recover your deployment after a disaster.

For information about analytics backup, restore, and disaster recover, refer to Backup, restore, and disaster recovery.

Previous topic: Preparing the Portal subsystem for disaster recovery on Cloud Pak for Integration v10.0.6.0

Recovering from a disaster on Cloud Pak for Integration v10.0.6.0

Install and configure API Connect, then restore the backed up data to recover the deployment after a disaster.

Before you begin

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
To successfully recover API Connect from a disaster, you must have previously completed the steps to prepare the Management, Portal, and Analytics subsystems for disaster recovery. Make sure that you can access the backups and deployment information because you cannot recover your deployment without it.

Attention: Successful disaster recovery depends on recovery of both the Management subsystem and the Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management subsystem first, and then immediately restore the Portal subsystem.

About this task

To recover from a disaster event, recreate your API Connect Cluster CR and add recovery details as explained in the following procedure. Use the file containing the original CR (which you saved during the preparation task) as a reference.

Note: Your recovery installation of Cloud Pak for Integration must include the Platform UI if your original deployment used it. In addition, the CR must use the same settings as in the original deployment.

Procedure

1. Prepare your Cloud Pak for Integration environment for deploying API Connect as explained in the appropriate version of the Cloud Pak for Integration documentation.
   After you prepare your environment, return to this procedure to configure the installation CR, deploy API Connect, and then restore data to the API Connect subsystems.

   Important: Do not install API Connect until you have configured the CR as explained in following steps.
2. Apply the Management and Portal encryption secrets to the cluster where you will install API Connect.
   a. Apply the YAML file that contains the Management database encryption secret into the cluster.
      Run the following command, using the filename of the encryption secret that you saved when you prepared the Management subsystem for disaster recovery:

      ```
      oc apply -f mgmt-enc-key.yaml
      ```

   b. Apply the YAML file that contains the Portal encryption secret into the cluster.
      Run the following command, using the filename of the encryption secret that you saved when you prepared the Portal subsystem for disaster recovery:

      ```
      oc apply -f portal-enc-key.yaml
      ```

3. Apply the YAML file that contains the CP4I credentials secret into the cluster.
   Run the following command, using the filename of the secret that you saved when you prepared the Management subsystem for disaster recovery:

   ```
   oc apply -f <cp4i_creds_secret>.yaml -n <APIC_namespace>
   ```

4. Set up the installation CR for deploying API Connect.
   In this step, you perform the initial steps for deploying API Connect in Cloud Pak for Integration, but you do not begin the actual installation until step 15. Instead, you will complete a series of steps to configure the CR with deployment settings and additional recovery settings.

   a. Log in to the IBM Cloud Pak Platform UI.
   b. On the home page, click Create capability.
   c. Select the API Connect tile and click Next.
   d. On the Create an instance of API Connect cluster page, select the deployment type and click Next.
      Choose the same deployment type that you used when you originally deployed API Connect.

   e. On the deployment settings page, click the YAML tab to edit the installation CR in YAML format.
   f. Copy the content of the saved CR into the YAML tab, replacing the default CR.
      Keep the YAML tab open. In the following steps, you will configure additional deployment and recovery settings in the new API Connect Cluster installation CR before you install API Connect.

5. Important: Make sure that your new version of the API Connect Cluster installation CR uses the same value for the `name` setting as the original CR.
6. Verify the backup and restore settings for each subsystem.
   For more information on the backup settings, see the following topics for instructions on configuring backup and restore settings:

   - Configuring backup settings for a fresh install of the Management subsystem on OpenShift or Cloud Pak for Integration v10.0.6.0
   - Configuring backups for Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0
   - Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration
7. Generate the Kubernetes secret for each subsystem's backups by running the appropriate command:
   For each secret, use the same name that you used in the original deployment (you can see the name in the `credentials` setting of each subsystem's backup section).

   - Management subsystem:
     - S3:

       ```
       oc create secret generic mgmt-backup-secret --from-literal=username='<Your_access_key-or-user_name>'
       --from-literal=password='<Your_access_key_secret-or-password>' -n <APIC_namespace>
       ```

     - SFTP with username and password:

       ```
       oc create secret generic mgmt-backup-secret --from-literal=username='<Your_user_name>'
       --from-literal=password='<Your_password>' -n <APIC_namespace>
       ```

     - SFTP with username and SSH-key

       ```
       oc create secret generic mgmt-backup-secret --from-literal=username='<Your_user_name>'
       --from-file=ssh-privatekey='<Your_private_key_file>' -n <APIC_namespace>
       ```

   - Portal subsystem:
     - S3:

       ```
       oc create secret generic portal-backup-secret --from-literal=username='<Your_access_key-or-user_name>'
       --from-literal=password='<Your_access_key_secret-or-password>' -n <APIC_namespace>
       ```

     - SFTP with username and password:

       ```
       oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>'
       --from-literal=password='<Your_password>' -n <APIC_namespace>
       ```

- SFTP with username and SSH-key

  ```
  oc create secret generic portal-backup-secret --from-literal=username='<Your_user_name>'
  --from-file=ssh-privatekey='<Your_private_key_file>' -n <APIC_namespace>
  ```

- Analytics:

  ```
  oc create secret generic analytics-backup-secret --from-literal=access_key='<Your_Access_Key>' --from-
  literal=secret_key='<Your_access_key_secret>' -n <APIC_namespace>
  ```

8. Add the saved Management and Portal encryption secrets to the installation CR.
   Add each secret to the appropriate **spec.*subsystem*** section of the CR, as shown in the following example:

   ```
   spec:
     management:
       encryptionSecret:
         secretName: mgmt-enc-key
     portal:
       encryptionSecret:
         secretName: portal-enc-key
   ```

9. Apply each of the Management client application credential secrets to the cluster.
   Run the following command to apply each secret, using the filename of each client application credential secret that you saved when you prepared the Management subsystem for disaster recovery:

   ```
   oc apply -f <secretName>.yaml
   ```

   For example, a deployment might use the following names for the secrets:

   ```
   atmCredentialSecret: management-atm-cred
   consumerToolkitCredentialSecret: management-ccli-cred
   consumerUICredentialSecret: management-cui-cred
   designerCredentialSecret: management-dsgr-cred
   governanceCredentialSecret: management-governance-cred
   juhuCredentialSecret: management-juhu-cred
   toolkitCredentialSecret: management-cli-cred
   uiCredentialSecret: management-ui-cred
   ```

10. Add each of the credential secrets to the **spec.management** section of installation CR.
    For example:

    ```
    spec:
      management:
        customApplicationCredentials:
        - name: atm-cred
          secretName: management-atm-cred
        - name: ccli-cred
          secretName: management-ccli-cred
        - name: cui-cred
          secretName: management-cui-cred
        - name: dsgr-cred
          secretName: management-dsgr-cred
        - name: governance-cred
          secretName: management-governance-cred
        - name: juhu-cred
          secretName: management-juhu-cred
        - name: cli-cred
          secretName: management-cli-cred
        - name: ui-cred
          secretName: management-ui-cred
    ```

11. Add the **siteName** property to the **spec.management** and **spec.portal** sections of the installation CR.
    For example, if the **a2a5e6e2** is the original **siteName** of the Management subsystem, and **890772e3** is the original **siteName** of the Portal subsystem, the CR looks like the following example:

    ```
    spec:
      management:
        siteName: "a2a5e6e2"
      portal:
        siteName: "890772e3"
    ```

12. If your original CR **name** was 10 characters or longer, you must add the **originalUID** or **metadata.uid** value from the original Management and Portal subsystems into the appropriate sections of the installation CR.
    Important: If your original CR name was 10 characters or longer and you do not add this value, the deployments and routes that are generated during installation will include a different UID, which will cause the recovery process to fail.
    The IDs were created with the original deployment of the Management and Portal subsystems, and the recovered subsystems must continue to use the same IDs. You can locate the ID in the Management (*apic-cluster-name*-mgt.yaml) or Portal (*apic-cluster-name*-ptl.yaml) CR that you saved while preparing for disaster recovery. (Do not use the **metadata.uid** setting from the API Connect Cluster CR).

    The CR setting that contains the ID depends on the version of API Connect that you are recovering:

    - Version 10.0.3 or later: **spec.originalUID**
    - Version 10.0.2 or earlier: **metadata.uid**

    Locate each of the original settings in the appropriate subsystem CR, and copy them to corresponding sections of the new API Connect Cluster CR using the same key and value.

    For example, if you originally deployed version 10.0.2, the setting name was **uid**. If you are deploying Version 10.0.3 as part of disaster recover, copy the original value and use it for the **originalUID** setting in the new CR. For example, in 10.0.3:

```
spec:
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
  portal:
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
```

13. If you installed the API Connect using an S3 provider for the Management subsystem backups, add the following annotation to the `apiconnectcluster` CR. If you specified SFTP or local as the backup, skip this step.

    - Using the Platform UI:
      - Edit the API Management instance and click the YAML tab to edit the CR.
      - Add the following statement to the `spec.metadata.annotations` section:

        ```
        apiconnect-operator/deployment-mode: disasterRecovery
        ```

        For example:

        ```
        metadata:
          annotations:
            apiconnect-operator/deployment-mode: disasterRecovery
        ```

    - Using the CLI:
      - Get the name of the CR by running the following command:

        ```
        oc get apiconnectcluster -o name -n <APIC_namespace>
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name
        ```

      - Add the following annotation to the `spec.metadata.annotations` section of the CR by running the following command:

        ```
        oc annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode="disasterRecovery" -n
        <APIC_namespace>
        ```

        The response looks like the following example:

        ```
        apiconnectcluster.apiconnect.domain/instance_name annotated
        ```

14. Verify that you correctly added all of the settings to the installation CR.
    For example, the completed CR might look like the following example:

```
apiVersion: apiconnect.ibm.com/v1beta1
kind: APIConnectCluster
metadata:
  namespace: apiconnect
  name: apis-minimum
  labels:
    app.kubernetes.io/instance: apiconnect
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: apiconnect-minimum
spec:
  license:
    accept: true
    license: L-RJON-C2YLGB
    metric: VIRTUAL_PROCESSOR_CORE
    use: nonproduction
  management:
    originalUID: "fa0f6f49-b931-4472-b84d-0922a9a92dfd"
    encryptionSecret:
      secretName: apis-minim-faaa44bf-enc-key
    siteName: "faaa44bf"
    customApplicationCredentials:
      - name: atm-cred
        secretName: apis-minim-faaa44bf-atm-cred
      - name: ccli-cred
        secretName: apis-minim-faaa44bf-ccli-cred
      - name: cli-cred
        secretName: apis-minim-faaa44bf-cli-cred
      - name: cui-cred
        secretName: apis-minim-faaa44bf-cui-cred
      - name: dsgr-cred
        secretName: apis-minim-faaa44bf-dsgr-cred
      - name: governance-cred
        secretName: apis-minim-faaa44bf-governance-cred
      - name: juhu-cred
        secretName: apis-minim-faaa44bf-juhu-cred
      - name: ui-cred
        secretName: apis-minim-faaa44bf-ui-cred
    analytics:
      client: {}
      ingestion: {}
    apiManagerEndpoint: {}
    cloudManagerEndpoint: {}
    consumerAPIEndpoint: {}
    platformAPIEndpoint: {}
    portal:
      admin: {}
    databaseBackup:
      path: ocp-dr-mgmt
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      s3provider: ibm
      schedule: 0 1 * * *
      protocol: objstore
      credentials: mgmt-backup-secret
```

```
  analytics:
    storage:
      enabled: true
      type: unique
    databaseBackup:
      chunkSize: 1GB
      credentials: a7s-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: ocp-dr-a7s
      schedule: 0 1 * * *
  storageClassName: rook-ceph-block
  profile: n1xc7.m48
  portal:
    adminClientSubjectDN: ""
    portalAdminEndpoint: {}
    portalUIEndpoint: {}
    originalUID: "447ea4b3-9514-4a84-a34b-ce0b349838da"
    encryptionSecret:
      secretName: apis-minim-913edd20-enc-key
    siteName: "913edd20"
    portalBackup:
      credentials: portal-backup-secret
      host: s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard
      path: ocp-dr-portal
      protocol: objstore
      schedule: 0 1 * * *
  version: 10.0.3
```

Important: If you are restoring your deployment in a different data center, the endpoints that are used in your original deployment must be the same in your new data center. The Platform UI in Cloud Pak for Integration autogenerates the endpoints if they are left empty in the CR, therefore ensure that you explicitly set the endpoints in the CR to match what was set in your original data center. For example:

```
apiManagerEndpoint:
      annotations:
        cert-manager.io/issuer: prod-ingress-issuer
        haproxy.router.openshift.io/timeout: 240s
      hosts:
        - name: prod-api-manager.example.com
        secretName: prod-be787dd3-api-manager
```

15. On the deployment page, install API Connect by clicking Create.
16. Wait for **all** subsystems to be created.
    When the Management subsystem is installed, you might see **backup** job pods and **stanza-create** job pods in **Error** state; for example:

```
m1-82b290a2-postgres-stanza-create-4zcgz                  0/1     Error       0          35m
m1-82b290a2-postgres-full-sch-backup-2g9hm                0/1     Error       0          20m
```

    This is expected behavior, for the following reasons:

    - The **stanza-create** job normally expects buckets or subdirectories within buckets to be empty. However, since you configured the Management subsystem with your already-populated S3 bucket (where your backups exist), the job will enter the **Error** state.
    - Any scheduled or manual backups will enter the **Error** state. Although you configured the Management subsystem with your already-populated S3 bucket, the new database isn't yet configured to write backups into remote storage.
    - The configurator job will fail because the CP4I credentials secret that you manually restored does not match the value in the Management database. As a result, the state of new cluster is not "Ready" and will show "6/7". This error will be resolved when you restore the Management subsystem from the backup that you prepared earlier.

    The errors will not prevent a successful restore, so continue to the next step.

17. Restore the subsystems in the following sequence:
    Important: Make sure that you restore using the backup (locate the one with the correct backup name and ID) that you created when you prepared each subsystem for disaster recovery.
    a. Restore the Management subsystem.
       When you perform a restore, you must complete the restoration of the Management subsystem first. Verify that the restoration completed successfully and that the Management subsystem is Ready. When the Management subsystem is healthy, proceed to the next step and restore the Portal subsystem.

    b. Restore the Developer Portal subsystem.
       Use the restore type **all** to ensure that you restore the complete subsystem and all Portal sites.

    c. Backing up and restoring the Analytics database on OpenShift and Cloud Pak for Integration.
18. Force the configurator to run again.
    The cluster will still not be ready after restore at this stage because the configurator has yet to successfully complete (refer to point 12). In order for the configurator to run again, delete the associated job so that a new pod will start running:

    a. Run the following command to get the list of jobs:

       ```
       oc get jobs -n <APIC_namespace>
       ```

    b. Run the following command to determine the name of your API Connect instance:

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

    c. Run the following command to delete the configurator job:

       ```
       oc -n <APIC_namespace> delete job <instance_name>-configurator
       ```

19. Verify that the recovery was successful:
    a. Ensure that you can log in to the Cloud Manager UI.
    b. Verify that your provider organizations exist.
    c. Ensure that you can log in to each Developer portal.
    d. Ensure that the Analytics dashboard contains all of the analytics data that you preserved.

20. After the successful recovery, remove the annotation that you added to the `apiconnectcluster` CR in step 13.
    If you skipped step 13, then skip this step as well.

    - Using the Platform UI:
        - Edit the API Management instance and click the YAML tab to edit the CR.
        - Delete the following statement from the `spec.metadata.annotations` section:

          ```
          apiconnect-operator/deployment-mode: disasterRecovery
          ```

    - Using the CLI:
        - Get the name of the CR by running the following command:

          ```
          oc get apiconnectcluster -o name -n <APIC_namespace>
          ```

          The response looks like the following example:

          ```
          apiconnectcluster.apiconnect.domain/instance_name
          ```

        - Remove the annotation by running the following command, making sure to include the trailing "-" on `deployment-mode-` to indicate the removal:

          ```
          oc -n <APIC_namespace> annotate apiconnectcluster/instance_name apiconnect-operator/deployment-mode-
          ```

          The response looks like the following example:

          ```
          apiconnectcluster.apiconnect.domain/instance_name annotated
          ```

Maintaining a two data center deployment

How to maintain a two data center disaster recovery (DR) deployment on Kubernetes and OpenShift, including information about normal operation, failure handling, and recovery.

See the following topics for information about failure handling, backup, upgrade, and removing a 2DCDR deployment.

For general information about two data center disaster recovery in API Connect, see Two data center warm-standby deployment on Kubernetes and OpenShift.

- **Key concepts of 2DCDR and failure scenarios**
  Understand how API Connect 2DCDR works and what to do when a data center fails.
- **Failing over to the warm-standby**
  How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration, for API Connect v10.0.7 and later.
- **Failing over to the warm-standby V10.0.6.0**
  How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.
- **Backup and restore requirements for a 2DCDR deployment**
  Points to consider when you configure database backup and restore settings on a two data center disaster recovery (2DCDR) deployment.
- **Removing a two data center deployment**
  Convert your 2DCDR deployment to a stand-alone deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.
- **Verifying replication between data centers**
  How to verify that your API Connect data is replicating to your warm-standby data center.
- **Synchronizing the ingress-ca certificates between sites**
- **Troubleshooting a two data center deployment**
  What to check when a 2DCDR deployment is not starting or not replicating data.

Related concepts

- Two data center warm-standby deployment on Kubernetes and OpenShift

Key concepts of 2DCDR and failure scenarios

Understand how API Connect 2DCDR works and what to do when a data center fails.

This topic discusses the concepts around the operation and maintenance of an API Connect 2DCDR deployment. For information on the planning and installation of a 2DCDR deployment, see: Two data center warm-standby deployment on Kubernetes and OpenShift.

Failover

Failover is a manual operation that requires updating the active management, portal, or both subsystems to become warm-standby, and the warm-standby subsystems to become active. The typical situations for completing a failover operation are:

- Operational failover: Where your active API Connect subsystems and data center are functioning normally, but you want to switch data centers for operational reasons, or to test the failover process.
  In this scenario, the active management and portal subsystems are converted to warm-standby first, and then the original warm-standby subsystems are converted to active.

- System down failover: Where your active data center, or the API Connect subsystems in that data center are in a failed state, and you must failover to the warm-standby to restore your API Connect service.
  In this scenario, the subsystems in the warm-standby data center are converted to active, and the network connectivity to and from the failed active subsystems is disabled - until it is possible to convert them to warm-standby.

Important:

- In all scenarios, an active-active configuration must be avoided. An active-active configuration is where the API Connect subsystems in both data centers are configured as active. This situation is commonly known as a split-brain. An active-active configuration means that the subsystem databases in each data center diverge from each other, and two management subsystems are both attempting to manage the other API Connect subsystems.
- When the active management subsystem is set to warm-standby, all data is deleted from its database. The management data from the new active (what was the warm-standby) is replicated to the new warm-standby data center. Do not set the active data center to warm-standby until you confirm that data replication is working, see: Verifying replication between data centers, or you have backups of your management data.

Reverting to a stand-alone deployment

If you want to revert to a stand-alone API Connect deployment, before you start the process, ensure that the data center where you choose to keep API Connect is set to active.

- When API Connect is reverted to be a stand-alone deployment on the warm-standby data center, all management and portal data is deleted from that data center. It becomes an empty stand-alone deployment.
- When API Connect on the active data center is reverted to be a stand-alone deployment, all management and portal data remains.

Failure scenarios

The action that you should take when a failure occurs depends on the type of failure and which data center the failure occurs in:

API Connect failure in the active data center.
When the active data center is still working correctly, and still has connectivity to the warm-standby data center, but API Connect is not working correctly:

1. It might be possible to recover API Connect on the active data center. Review the troubleshooting documentation:
   - Troubleshooting two data center replication problems.
   - Installation troubleshooting.
   - Troubleshooting operational problems.
2. Follow the failover steps to make the warm-standby the active data center. If your failed data center cannot be changed to warm-standby, then disable network connectivity to and from it, to avoid an active-active situation if the data center recovers unexpectedly.
3. Gather API Connect post-mortem logs from the failed data center and open a support request: Gathering post-mortem logs.
4. Do not restore the network connectivity to and from the API Connect subsystems in the failed data center until they are set to warm-standby.

Complete active data center failure
When the active data center has a failure, and API Connect is not accessible, follow these steps:

1. Disable the network connections from and to your active subsystems. Network disablement is required to prevent an active-active scenario if your failed data center recovers before you can change it to warm-standby.
2. Follow the failover steps to convert the warm-standby data center to active: Failing over to the warm-standby.
3. Do not restore the network connectivity to and from the API Connect subsystems in the failed data center until they are set to warm-standby.

Network failure that prevents API calls or user access to Management and Portal subsystems on active data center.
In this scenario, the API Connect 2DCDR deployment is working correctly, but unusable because the active data center is not accessible for API calls or for Management and Portal UI/CLI/REST API users.

1. Complete a failover, setting warm-standby to active, and the active to warm-standby: Failing over to the warm-standby.

API Connect failure on the Warm-standby data center
Where the API Connect pods on the warm-standby data center are not running as expected.

1. Review the troubleshooting documentation Troubleshooting two data center replication problems.
2. Gather API Connect post-mortem logs from the failed data center and open a support case: Gathering post-mortem logs.

Warm-standby data center failure
The data center where API Connect was running as warm-standby fails. In this case when the data center is recovered:

1. Verify that API Connect is running correctly as warm-standby and that the data from the active is replicated to it: Verifying replication between data centers.

Network failure between data centers
When the network connection between data centers is recovered, verify that 2DCDR database replication resumes: Verifying replication between data centers.

Management 2DCDR operational modes and statuses

The management CR `status` section includes information on the current 2DCDR mode of the management subsystem, and its status. The status section is included in the output of

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

```
...
status:
...
  haMode: active
  ...
  haStatus:
```

Table 1. 2DCDR Management subsystem operational modes (`status.haMode`)

Operational mode	Description
`active`	The subsystem pods are ready and in the correct disaster recovery state for the active data center.
`passive`	The subsystem pods are ready and in the correct disaster recovery state for the warm-standby data center.
`standalone`	The management subsystem is not configured for 2DCDR.

Table 2. Management subsystem 2DCDR statuses (`status.haStatus`)

status.haStatus	Description

status.haStatus	Description
Pending	2DCDR initial state.
Warning	Replication is failing.
Ready	Replication is running.
Error	Error in 2DCDR configuration.
PeerNotReachable	The subsystem cannot contact the subsystem on the other site, and so replication is failing.
BlockedWarmStandbyConversion	The warm-standby subsystem is in `Blocked` state, waiting for other site to become active. This state occurs when user converts existing warm-standby to active.
PromotionBlocked	Current warm-standby site promotion to active is blocked
ReadyForPromotion	Current warm-standby site is ready for promotion

Portal 2DCDR operational states

The portal subsystem has various transitory states that it passes through during startup and failover. The current 2DCDR operational state can be observed from the CR status section, for example:

```
kubectl get ptl -o yaml
```

```
...
status:
...
  haMode: active
  ...
  haStatus:
```

Table 3. 2DCDR Portal subsystem operational states

Operational state	Description
`progressing to active`	Pods are progressing to the active state, but the subsystem is not capable of serving traffic.
`progressing to active (ready for traffic)`	At least one pod of each type is ready for traffic. The dynamic router can be linked to this service.
`active`	The subsystem pods are ready and in the correct disaster recovery state for the active data center.
`progressing to passive`	Pods are progressing to the warm-standby state. The subsystem is not capable of serving traffic.
`progressing to passive (ready for traffic)`	At least one pod of each type is ready for traffic.
`passive`	The subsystems are ready and in the correct disaster recovery state for the warm-standby data center.
`progressing to down`	Portal is preparing to do a factory reset. This happens to the warm-standby portal when the `multiSiteHA` section is removed.

Failing over to the warm-standby

How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration, for API Connect v10.0.7 and later.

Before you begin

This topic describes the failover operation for API Connect v10.0.7 and later. If you are on v10.0.6, then refer to Failing over to the warm-standby V10.0.6.0.

Ensure that you have read and understand the 2DCDR concepts and reviewed the failure scenarios that are described in Key concepts of 2DCDR and failure scenarios. Do not proceed with the failover operation until you confirm that failover the correct course of action for your situation.

About this task

Be aware of the following important points:

- The first step in the process is to convert the active data center to warm-standby. When the active data center is converted to warm-standby, all data is deleted from the active data center's management database, to be replaced by data that is copied from the warm-standby when it becomes active.
  Do not proceed with failover if you suspect the warm-standby data center also has problems, and you are unsure it has the most recent management data. See Verifying replication between data centers, and consider restoring your active site from backup instead of attempting a failover: Backup and restore requirements for a 2DCDR deployment.
- In all scenarios, an active-active configuration must be avoided. An active-active configuration is where the API Connect subsystems in both data centers are configured as active. This situation is commonly known as a split-brain. An active-active configuration means that the subsystem databases in each data center diverge from each other, and two management subsystems are both attempting to manage the other API Connect subsystems.
- If the active data center failure prevents you from converting it to warm-standby, then you must disable the network connectivity to and from the failed management and portal subsystems on this data center, to prevent an accidental active-active situation if your failed data center recovers unexpectedly.
- If you are doing an operational failover, the process causes a temporary management and portal UI outage, until the new warm-standby completes the conversion to active.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the multiSiteHA section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Procedure

- **Management subsystem failover**
  In these example steps, DC1 is the active data center and DC2 is the warm-standby data center.

To complete a failover from DC1 to DC2, you first set DC1 to warm-standby before you set DC2 to active. This order is required to prevent an active-active situation.

1. Set the DC1 management subsystem to be warm-standby.
   - On Kubernetes and OpenShift individual subsystem CR deployments: Create a file called demote-to-warm-standby.yaml, and paste in the following text:

     ```
     metadata:
       annotations:
         apiconnect-operator/dr-data-deletion-confirmation: "true"
     spec:
       multiSiteHA:
         mode: passive
     ```

     Apply the updates that are specified in the demote-to-warm-standby.yaml file to your management CR:

     ```
     kubectl patch mgmt <mgmt cr name> --type merge --patch "$(cat demote-to-warm-standby.yaml)"
     ```

   - On Cloud Pak for Integration and OpenShift top-level CR deployments: Create a file called demote-to-warm-standby.yaml, and paste in the following text:

     ```
     metadata:
       annotations:
         apiconnect-operator/dr-data-deletion-confirmation: "true"
     spec:
       management:
         multiSiteHA:
           mode: passive
     ```

     Apply the updates that are specified in the demote-to-warm-standby.yaml file to your API Connect CR:

     ```
     oc patch apiconnectcluster <apic cr name> --type merge --patch "$(cat demote-to-warm-standby.yaml)"
     ```

   Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network connectivity to and from the management subsystem in DC1 is disabled, to prevent an accidental active-active situation if DC1 recovers unexpectedly. Do not restore network connectivity until DC1 is set to warm-standby.
2. Run the following command on DC1 to check that the management subsystem HA mode is no longer `active`:

   ```
   kubectl get mgmt -o yaml
   ```

   During transition to warm-standby, the `status.haMode` shows `BlockedWarmStandbyConversion`. You can monitor the transition with:

   ```
   kubectl get mgmt -o=jsonpath='{"Status: "}{.status.phase}{"\n - HA Mode: "}{.status.haMode} {"\n - Ha Status: "}
   {.status.haStatus[?(@.status=="True")].type}{"\n"}'
   ```

   ```
   Status: Blocked
    - HA Mode: BlockedWarmStandbyConversion
    - Ha Status: BlockedWarmStandbyConversion
   ```

3. Change the DC2 management subsystem from warm-standby to active.
   Note: If API Connect is installed as part of Cloud Pak for Integration, then before you convert your warm-standby to active, complete the following checks:
   a. Check whether a `configurator` job exists:

      ```
      oc -n <namespace> get jobs -l "app.kubernetes.io/managed-by=apiconnect-operator" | grep configurator
      ```

      if a configurator job exists, then delete it:

      ```
      oc -n <namespace> delete job <configurator job name>
      ```

   b. Remove the `spec.disabledServicesproperty.configurator` property from your `APIConnectCluster` CR. You added this property during installation step 4 of Installing API Connect on the warm-standby data center

   Check the DC2 management subsystem `haStatus`:

   ```
   kubectl get mgmt -o yaml
   ```

   ```
   status
     ...
     haMode: ReadyForPromotion
       haStatus:
         ...
         type: ReadyForPromotion
     ...
   ```

   wait until the `haStatus` shows `ReadyForPromotion` before proceeding. You can monitor the transition with:

   ```
   kubectl get mgmt -o=jsonpath='{"Status: "}{.status.phase}{"\n - HA Mode: "}{.status.haMode} {"\n - Ha Status: "}
   {.status.haStatus[?(@.status=="True")].type}{"\n"}'
   ```

   ```
   Status: Blocked
    - HA Mode: BlockedWarmStandbyConversion
    - Ha Status: BlockedWarmStandbyConversion
   ```

   - On Kubernetes and OpenShift individual subsystem CR deployments:
     Edit the DC2 management CR:

     ```
     kubectl edit mgmt
     ```

     Change `spec.multiSiteHA.mode` to `active`:
   - On Cloud Pak for Integration and OpenShift top-level CR deployments:
     Edit the DC2 `APIConnectCluster` CR:

     ```
     oc edit apiconnectcluster
     ```

     Change `spec.management.multiSiteHA.mode` to `active`.

4. Run the following command on both data centers to confirm when management subsystem failover is complete:

   ```
   kubectl get mgmt
   ```

   Example output:

   ```
   production-mgmt   n/n   Running   10.0.x.y   10.0.x.y        Management is ready. HA status Ready - see HAStatus in
   CR for details
   ```

   Note: Failover can take 10 minutes or more to complete. If failover does not complete, you can check the API Connect operator pod log for errors (search for the word "Multi").
5. Update your dynamic router to redirect all traffic to DC2 instead of DC1. Until the DC2 site becomes `active`, the UIs might not be operational.

- **Developer Portal subsystem failover**
  In these example steps, DC1 is the active data center and DC2 is the warm-standby data center.

  If you have multiple Developer Portal services, you must repeat these steps for each Developer Portal subsystem that you want to failover.

  1. Set the DC1 Developer Portal subsystem to be warm-standby.
     - On Kubernetes and OpenShift individual subsystem CR deployments:
       Edit the DC1 `PortalCluster` CR:

       ```
       kubectl edit ptl
       ```

       Change `multiSiteHA.mode` to `passive`.
     - On Cloud Pak for Integration and OpenShift top-level CR deployments:
       Edit the DC1 `APIConnectCluster` CR:

       ```
       oc edit apiconnectcluster
       ```

       Change `spec.portal.multiSiteHA.mode` to `passive`.

     Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network connectivity to and from the portal subsystem in DC1 is disabled, to prevent an accidental active-active situation if DC1 recovers unexpectedly. Do not restore network connectivity until DC1 is set to warm-standby.
  2. Run the following command on DC1 to check the Developer Portal subsystem status:

     ```
     kubectl get ptl -o yaml
     ```

     Continue to the next step when `status.haMode` is set to `progressing to passive`, or any of the `passive` states.
     For example:

     ```
     status:
       ...
       haMode:                              progressing to passive
       ...
     ```

  3. Change the DC2 Developer Portal subsystem from warm-standby to active.
     - On Kubernetes and OpenShift individual subsystem CR deployments:
       Edit the DC2 `PortalCluster` CR:

       ```
       kubectl edit ptl
       ```

       Change `spec.multiSiteHA.mode` to `active`.
     - On Cloud Pak for Integration and OpenShift top-level CR deployments:
       Edit the DC2 `APIConnectCluster` CR:

       ```
       oc edit apiconnectcluster
       ```

       Change `spec.portal.multiSiteHA.mode` to `active`.
  4. Update your dynamic router to redirect all traffic to DC2 instead of DC1.
  5. Run the following command to check that the Developer Portal services on DC1 and DC2 are ready for traffic:

     ```
     kubectl get ptl -o yaml
     ```

     The services are ready for traffic when the `status.haMode` is set to `passive` on DC1, and is set to `active` on DC2.

- **Failover of both management and portal subsystems**
  Failover the management subsystem first, followed by the portal subsystem.

  How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. Approximate timings are:

  For the management subsystem:
  - `warm-standby` to `active`: 5 minutes
  - `active` to `warm-standby`: 15 minutes

  For the Developer Portal:
  - `warm-standby` to `active`: 15 - 40 minutes
  - `active` to `warm-standby`: 10 minutes

What to do next

If the original active data center was successfully converted to warm-standby, then verify that replication is working: Verifying replication between data centers. If replication is working, you can either:

- Revert your 2DCDR deployment to the original active and warm-standby data center designations. To revert your deployment, follow the same failover steps in this topic.
- Do nothing, and continue with your current active and warm-standby data center designations.

If your failed data center cannot be updated to warm-standby, then ensure that the network links to and from your management and portal subsystems in the failed data center are disabled. If the network links remain enabled, then an accidental active-active situation might occur if your failed data center recovers unexpectedly.

When you are able to recover the failed data center, ensure that the management and portal subsystems are set to warm-standby before you restore the network connectivity to prevent an active-active situation.
Note: The longer your original active data center was in a failed state, the longer data replication takes when it is recovered to warm-standby.
If you expect your failed data center to be down for a long time, then convert your active data center to a stand-alone deployment. See Removing a two data center deployment.

Failing over to the warm-standby V10.0.6.0

How to complete service failover in a 2DCDR deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.

Before you begin

Note: This topic applies to V10.0.6.0. For failover steps on V10.0.7.0 and later, see Failing over to the warm-standby.
Ensure that you have read and understand the 2DCDR concepts and reviewed the failure scenarios that are described in Key concepts of 2DCDR and failure scenarios. Do not proceed with the failover until you confirm that it is the correct course of action for your situation.

About this task

Carefully follow the steps in this topic. Be aware of the following important points:

- The first step in the process is to set the active data center to warm-standby. When the active data center is set to warm-standby, all data is deleted from the active data center's management database, to be replaced by data copied from the warm-standby when it becomes active. Do not proceed with failover if you suspect there is also a problem on the warm-standby data center and you are unsure it has the most recent management data. See Verifying replication between data centers, and consider restoring your active site from backup instead of attempting a failover: Backup and restore requirements for a 2DCDR deployment.
- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center. In this case, you should revert your deployment to a single data center topology. For more information, see Removing a two data center deployment.
- Avoid both data centers being set to active at the same time while the network links between data centers are enabled. This causes split-brain.
- If the active data center failure prevents you from updating the active data center API Connect custom resources (CRs), then you should disable the network links used by API Connect between the data centers.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the multiSiteHA section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Procedure

- **Management subsystem failover. In this example, DC1 is the active and DC2 is the warm-standby.**
  To initiate a failover from DC1 to DC2, you first must set DC1 to warm-standby before you set the DC2 data center to active. This action is needed to prevent split-brain, as there can't be two active data centers at the same time. The following instructions show how to failover DC1 to DC2 for the Management subsystem.

  1. Set the DC1 Management subsystem to be warm-standby.
     Edit the DC1 `ManagementCluster` custom resource (CR) by running the following command:

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

     and change the `mode` to `passive`:

     ```
     multiSiteHA:
       mode: passive
     ```

     Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network links between DC1 and DC2 are disabled before you set DC2 to be active. You must then not restore the network links until you can set DC1 to be warm-standby.
  2. Run the following command on DC1 to check that the Management subsystem HA mode is no longer `active`:

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

     During transition to warm-standby, the `Ha mode` part of the `Status` output shows `progressing to passive`. When the transition is complete, it shows `passive`.

     ```
     Status:
       ...
       Ha mode:                              passive
       ...
     ```

     You can proceed to set DC2 to active while DC1 is still `progressing to passive`.

  3. Change the DC2 Management subsystem from warm-standby to active.
     Note: If API Connect is installed as part of Cloud Pak for Integration, then before you convert your warm-standby to active, complete the following checks:
     a. Check whether a `configurator` job exists:

        ```
        oc -n <namespace> get jobs -l "app.kubernetes.io/managed-by=apiconnect-operator" | grep configurator
        ```

        if a configurator job exists, then delete it:

        ```
        oc -n <namespace> delete job <configurator job name>
        ```

     b. Remove the `spec.disabledServicesproperty.configurator` property from your `APIConnectCluster` CR. You added this property during installation step 4 of Installing API Connect on the warm-standby data center

     Edit the DC2 `ManagementCluster` custom resource (CR) by running the following command:

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

and change the `mode` to `active`:

```
multiSiteHA:
  mode: active
```

4. Run the following command on DC2 to check that the Management subsystem is ready for traffic:

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

The services are ready for traffic when the `Ha mode` part of the `Status` object is set on DC2 to `active`. For example:

```
Status:
  ...
  Ha mode:                              active
  ...
```

Note: Failover can take 10 minutes or more to complete. If there are any problems, you can check the API Connect operator pod log for errors (search for the word "Multi").

5. Update your dynamic router to redirect all traffic to DC2 instead of DC1. Until the DC2 site becomes `active`, the UIs might not be operational.

- **Developer Portal subsystem failover**
If you have multiple portal services, you can failover a specific portal service, or all of the portal services that are running in a particular data center.

The following instructions show how to failover DC1 to DC2 for the Developer Portal subsystem. If you have multiple Developer Portal services, you must repeat these steps for each Developer Portal subsystem that you want to failover.

Note:
  - Developer Portal failover does result in a temporary Developer Portal website outage until the warm-standby data center is ready to accept traffic.
  - If you want to monitor the failover process, you can run the following command to check the status:

```
kubectl describe ptl
```

  - Do not edit the portal CR file during the failover process.

1. Set the DC1 Developer Portal subsystem to be warm-standby.
Edit the DC1 `PortalCluster` custom resource (CR) by running the following command:

```
kubectl edit ptl <ptl-cluster-name> -n <namespace>
```

and change the `mode` to `passive`:

```
multiSiteHA:
  mode: passive
```

Where *<ptl-cluster-name>* is the name of your portal cluster if you have more than one. To see a list of your portal clusters, run:

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

Note: If API Connect on DC1 is down, such that you can't set it to be warm-standby, you must ensure that the network links between DC1 and DC2 are disabled before you set DC2 to be active. You must then not restore the network links until you can set DC1 to be warm-standby.

2. Run the following command on DC1 to check the Developer Portal subsystem status:

```
kubectl describe ptl <ptl-cluster-name> -n <namespace>
```

You can continue to the next step when the `Ha mode` part of the `Status` object is set to `progressing to passive`, or any of the `passive` states. For example:

```
Status:
  ...
  Ha mode:                              progressing to passive
  ...
```

Important: You must not set DC2 to active until DC1 is either in `passive` or `progressing to passive` state. In other words, DC1 and DC2 must never both be in active state at the same time.

3. Change the DC2 Developer Portal subsystem from warm-standby to active.
Edit the DC2 `PortalCluster` custom resource (CR) by running the following command:

```
kubectl edit ptl <ptl-cluster-name> -n <namespace>
```

and change the `mode` to `active`:

```
multiSiteHA:
  mode: active
```

Where *<ptl-cluster-name>* is the name of your portal cluster if you have more than one. To see a list of your portal clusters, run:

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

4. Update your dynamic router to redirect all traffic to DC2 instead of DC1.
5. Run the following command to check that the Developer Portal services on DC1 and DC2 are ready for traffic:

```
kubectl describe ptl <ptl-cluster-name> -n <namespace>
```

The services are ready for traffic when the `Ha mode` part of the `Status` object is set on DC1 to `passive`, and is set on DC2 to `active`. For example, on DC2:

```
Status:
  ...
  Ha mode:                              active
  ...
```

- **Failover of both Management and Portal subsystems**
Failover the Management subsystem first, followed by the Portal subsystem.

How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. Approximate timings are:

For the Management subsystem:
- `warm-standby` to `active`: 5 minutes
- `active` to `warm-standby`: 15 minutes

For the Developer Portal:
- `warm-standby` to `active`: 15 - 40 minutes
- `active` to `warm-standby`: 10 minutes

What to do next

If the original active data center was successfully converted to warm-standby, then verify that replication is working: Verifying replication between data centers. If replication is working, you can either:

- Revert your 2DCDR deployment to the original active and warm-standby data center designations. To revert your deployment, follow the same failover steps in this topic.
- Do nothing, and continue with your current active and warm-standby data center designations.

If your failed data center cannot be updated to warm-standby, then ensure that the network links to and from your management and portal subsystems in the failed data center are disabled. If the network links remain enabled, then an accidental active-active situation might occur if your failed data center recovers unexpectedly.

When you are able to recover the failed data center, ensure that the management and portal subsystems are set to warm-standby before you restore the network connectivity to prevent an active-active situation.
Note: The longer your original active data center was in a failed state, the longer data replication takes when it is recovered to warm-standby.
If you expect your failed data center to be down for a long time, then convert your active data center to a stand-alone deployment. See Removing a two data center deployment.

Backup and restore requirements for a 2DCDR deployment

Points to consider when you configure database backup and restore settings on a two data center disaster recovery (2DCDR) deployment.

In a 2DCDR deployment, the backup requirements are different for the portal and management databases.

Management database 2DCDR backup and restore requirements

- Different remote backup locations must be specified for each data center.
- Both active and warm-standby management subsystems run the scheduled database backup job.
- For manual database backups, always run the warm-standby backup first.
- For scheduled database backups, configure the warm-standby's backup to run one minute before the active site's backup.
- V10.0.6 only: Disable 2DCDR before you change the configuration of S3 backups:
  1. Remove the `multiSiteHA` section from the `spec` section of the management CRs on both data centers. Take note of which data center is the active and which is the warm-standby. Keep a copy of the `multiSiteHA` sections that you remove.
  2. Configure your S3 backup settings. You must specify different backup locations for each data center, see Backup and restore requirements for a 2DCDR deployment.
  3. Add the `multiSiteHA` sections back to the management CRs in both data centers, ensuring that your original active is set to be the active again.

  You do not need to disable 2DCDR for SFTP or local backups.

- V10.0.6 only: Do not restore the management database backup from one data center, to the other data center.
- To restore the management database, you must convert both sites to standalone (non-2DCDR) before you can run the restore process.
- To restore a management database backup, follow these steps:
  1. Make each management data center a stand-alone API Connect deployment: Temporary conversion to stand-alone.
  2. Restore the database backup to your previously active data center: Restoring from a management database backup.
  3. Restore the 2DCDR configuration to the management CR in the restored data center. Ensure that the `multiSiteHA` section defines the data center as the active.
  4. Restore the 2DCDR configuration to the other data center. Ensure that the `multiSiteHA` section defines the data center as the warm-standby (passive).
  5. The management database takes some time to synchronize across the data centers. To confirm that replication is working, see: Verifying replication between data centers.

  Note: If you want to restore the backup to your previously warm-standby data center, and make that the active, then after step 1 follow these steps:
  1. To prevent an active-active situation, disable network connectivity to the management subsystem in your previously active data center.
  2. Restore the backup to your previously warm-standby data center.
  3. Restore the 2DCDR configuration to the management CR in the restored data center. Ensure that the `multiSiteHA` section defines the data center as the active.
  4. Restore the 2DCDR configuration to the management CR to the other data center. Ensure that the `multiSiteHA` section defines the data center as the warm-standby (passive).
  5. Enable network connectivity to your new warm-standby data center.
  6. The management database takes some time to synchronize across the data centers. To confirm that replication is working, see: Verifying replication between data centers.

Portal subsystem 2DCDR backup requirements

- The backup settings for both the active and warm-standby portal must be identical.
- Only the active data center portal takes backups. On failover, the backup job of the new warm-standby is disabled, and the backup job of the new active is enabled.
- Restore portal database backups on the active data center.

Configuring database backups

The steps for configuring database backups for both subsystems are the same as for a single API Connect deployment, but keep the additional 2DCDR requirements in mind when following the steps in Backing up, restoring, and disaster recovery.

Related concepts

- Two data center warm-standby deployment on Kubernetes and OpenShift

Removing a two data center deployment

Convert your 2DCDR deployment to a stand-alone deployment on Kubernetes, OpenShift, and Cloud Pak for Integration.

Ensure that you have read and understand the concepts of 2DCDR. See Two data center warm-standby deployment on Kubernetes and OpenShift and Key concepts of 2DCDR and failure scenarios.
Note: When you convert the warm-standby data center to stand-alone, all data is deleted from its management database. Data deletion is to prevent an active-active situation, where the warm-standby attempts to communicate with existing portal, gateway, and analytics subsystems that the formally active data center is also communicating with.
Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the multiSiteHA section for the subsystem in the top-level CR, instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the top-level CR.

Temporary conversion to stand-alone

Some API Connect maintenance operations, such as management database restore require that 2DCDR is temporarily disabled.

To convert a 2DCDR management or portal subsystem to stand-alone, follow these steps:

1. Take a backup of your current 2DCDR configuration, by backing up the subsystem CRs.
   a. On Kubernetes and OpenShift where individual subsystem CRs are used, take the management subsystem CR backup with:

      ```
      kubectl get mgmt -o yaml > mgmt_<dr status>_2dcdr_backup.yaml
      ```

      and take the portal CR backup with:

      ```
      kubectl get ptl -o yaml > portal_<dr status>_2dcdr_backup.yaml
      ```

      where *<dr status>* is the current 2DCDR status of the subsystem, either active or warm-standby.
   b. On Cloud Pak for Integration and OpenShift where the top-level CR is used, run:

      ```
      kubectl get apiconnectcluster -o yaml > apic_<dr status>_2dcdr_backup.yaml
      ```

      where *<dr status>* is the current 2DCDR status of the subsystem, either active or warm-standby. The `APIConnectCluster` CR contains the 2DCDR settings for both management and portal subsystems.
2. Remove the 2DCDR configuration from the management subsystem. The steps to take are different for active and warm-standby:
   a. Active management subsystem:
      i. On Kubernetes and OpenShift where individual subsystem CRs are used, run:

      ```
      kubectl patch managementcluster <mgmt cr name> --type json -p='[{"op": "remove", "path": "/spec/multiSiteHA"}]'
      ```

      ii. On Cloud Pak for Integration and OpenShift where the top-level CR is used:

      ```
      kubectl patch apiconnectcluster <apic cr name> --type json -p='[{"op": "remove", "path":
      "/spec/management/multiSiteHA"}]'
      ```

   b. Warm-standby management subsystem:
      To convert the management subsystem from warm-standby to stand-alone, you must acknowledge acceptance of this in an annotation in the management CR. The steps to convert warm-standby management to stand-alone are:
      i. On Kubernetes and OpenShift where individual subsystem CRs are used, create a file that is called remove-2dcdr-config-from-warm-standby.yaml, and paste in the following text:

      ```
      metadata:
        annotations:
          apiconnect-operator/dr-data-deletion-confirmation: "true"
      spec:
        multiSiteHA: null
      ```

      Apply the file to your management CR with:

      ```
      kubectl patch managementcluster <mgmt cr name> --type merge --patch "$(cat remove-2dcdr-config-from-warm-
      standby.yaml)"
      ```

      ii. On Cloud Pak for Integration and OpenShift where the top-level CR is used, create a file that is called remove-2dcdr-config-from-warm-standby.yaml and paste in the following text:

      ```
      metadata:
        annotations:
          apiconnect-operator/dr-data-deletion-confirmation: "true"
      spec:
        management:
          multiSiteHA: null
      ```

      Apply the file to your top-level CR with:

      ```
      kubectl patch apiconnectcluster <mgmt cr name> --type merge --patch "$(cat remove-2dcdr-config-from-warm-
      standby.yaml)"
      ```

3. Remove the 2DCDR configuration from the portal subsystem:
   a. On Kubernetes and OpenShift where individual subsystem CRs are used, run:

      ```
      kubectl patch portalcluster <portal CR name> --type json -p='[{"op": "remove", "path": "/spec/multiSiteHA"}]'
      ```

      Delete the `spec.multiSiteHA` section.
   b. On Cloud Pak for Integration and OpenShift where the top-level CR is used, run:

      ```
      kubectl patch apiconnectcluster <apic cr name> --type json -p='[{"op": "remove", "path":
      "/spec/portal/multiSiteHA"}]'
      ```

4. Complete the maintenance operation that required the temporary conversion to stand-alone, such as management database restore.
5. Restore the 2DCDR configuration by adding back the `multiSiteHA` sections to your subsystem CRs. You can restore these sections by copying the `multiSiteHA` sections from the backup files you took in step 1, and pasting them into the corresponding CRs:
   a. On Kubernetes and OpenShift where individual subsystem CRs are used:
      i. Edit the management CR:

         ```
         kubectl edit mgmt
         ```

         and paste in the `multiSiteHA` section from your backup mgmt_*<dr status>*_2dcdr_backup.yaml.
      ii. Edit the portal CR:

         ```
         kubectl edit ptl
         ```

         and paste in the `multiSiteHA` section from your backup portal_*<dr status>*_2dcdr_backup.yaml.
   b. On Cloud Pak for Integration and OpenShift where the top-level CR is used, edit the top-level CR:

      ```
      kubectl edit apiconnectcluster
      ```

      and in the `spec.management.multiSiteHA`, paste in the contents from the corresponding `multiSiteHA` sections of your apic_*<dr status>*_2dcdr_backup.yaml

Permanently revert to stand-alone

If you want to revert to a single data center topology:

1. Decide which data center you want to keep. If this data center is not the current active data center, then complete the failover operation to make it the active: Failing over to the warm-standby.
2. Remove the 2DCDR configuration section from the management subsystem CR on the active data center.
   a. On Kubernetes and OpenShift where individual subsystem CRs are used, run:

      ```
      kubectl patch managementcluster <mgmt cr name> --type json -p='[{"op": "remove", "path": "/spec/multiSiteHA"}]'
      ```

   b. On Cloud Pak for Integration and OpenShift where the top-level CR is used, run:

      ```
      kubectl patch apiconnectcluster <apic cr name> --type json -p='[{"op": "remove", "path":
      "/spec/management/multiSiteHA"}]'
      ```

3. Remove the 2DCDR configuration from the portal subsystem:
   a. On Kubernetes and OpenShift where individual subsystem CRs are used, run:

      ```
      kubectl patch portalcluster <portal CR name> --type json -p='[{"op": "remove", "path": "/spec/multiSiteHA"}]'
      ```

      Delete the `spec.multiSiteHA` section.
   b. On Cloud Pak for Integration and OpenShift where the top-level CR is used, run:

      ```
      kubectl patch apiconnectcluster <apic cr name> --type json -p='[{"op": "remove", "path":
      "/spec/portal/multiSiteHA"}]'
      ```

4. The next action that you take depends on what you want to do with your warm-standby data center:
   a. If you want to permanently revert to a single data center topology, then uninstall API Connect on the warm-standby data center. See: Uninstalling API Connect.
   b. Leave your warm-standby as it is, in case you decide to revert your stand-alone data center to a 2DCDR active.
   c. Convert your warm-standby subsystems to stand-alone. This operation erases all data from the management and portal databases.
      To convert the management subsystem from warm-standby to stand-alone, you must acknowledge acceptance of this in an annotation in the management CR. The steps to convert warm-standby management to stand-alone are:
      i. On Kubernetes and OpenShift where individual subsystem CRs are used, create a file that is called remove-2dcdr-config-from-warm-standby.yaml, and paste in the following text:

         ```
         metadata:
           annotations:
             apiconnect-operator/dr-data-deletion-confirmation: "true"
         spec:
           multiSiteHA: null
         ```

         Apply the file to your management CR with:

         ```
         kubectl patch managementcluster <mgmt cr name> --type merge --patch "$(cat remove-2dcdr-config-from-warm-
         standby.yaml)"
         ```

      ii. On Cloud Pak for Integration and OpenShift where the top-level CR is used, create a file that is called remove-2dcdr-config-from-warm-standby.yaml and paste in the following text:

         ```
         metadata:
           annotations:
             apiconnect-operator/dr-data-deletion-confirmation: "true"
         spec:
           management:
             multiSiteHA: null
         ```

Apply the file to your top-level CR with:

```
kubectl patch apiconnectcluster <mgmt cr name> --type merge --patch "$(cat remove-2dcdr-config-from-warm-standby.yaml)"
```

To convert portal from warm-standby to stand-alone, manually remove the `spec.multiSiteHA` section from the portal CR:

i. On Kubernetes and OpenShift where individual subsystem CRs are used, run:

```
kubectl edit ptl
```

Delete the `spec.multiSiteHA` section.

ii. On Cloud Pak for Integration and OpenShift where the top-level CR is used, run:

```
kubectl edit apiconnectcluster
```

Delete the `spec.portal.multiSiteHA` section.

Verifying replication between data centers

How to verify that your API Connect data is replicating to your warm-standby data center.

The management and portal UIs are not accessible on the warm-standby data center, so it is not possible to check that data replication is working by logging in to the UIs. Instead, you can verify that the warm-standby management and portal databases are in sync with command-line operations.
Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Verifying Management replication

The management CR reports a warning if it detects a problem with management data replication:

```
kubectl get mgmt
```

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Warning   10.0.7.0-0   10.0.7.0-0           Management is ready. HA Status Warning - see HAStatus in CR
for details   8m59s
```

```
kubectl get mgmt -o yaml
```

```
...
status:
  haStatus
    {
      "lastTransitionTime": "2023-03-31T19:47:08Z",
      "message": "Replication not working",
      "reason": "na",
      "status": "True",
      "type": "Warning"
    }
```

If the warning persists, then see Troubleshooting a two data center deployment.
To verify replication by creating new data on the active and checking that it is copied to the warm-standby, follow these steps:

1. Login to the Cloud Manager UI.
2. Create a local user registry called `2dcdrtest`: Configuring a Local User Registry.
3. Run the following command in your warm-standby data center to query the management database for the new user registry:
   a. On v10.0.7 and later, run:

   ```
   kubectl exec -n <namespace> `kubectl -n <namespace> get po -l role=primary -o name` -- bash -c "psql -U postgres apim -c \"select name,created_at from user_registry where name = '2dcdrtest'\""
   ```

   b. On v10.0.6, run:

   ```
   kubectl exec -n <namespace> `kubectl -n <namespace> get po -l role=master -o name` -- bash -c "psql apim -c \"select name,created_at from user_registry where name = '2dcdrtest'\""
   ```

   Expected output is:

   ```
   Defaulted container "database" out of: database, set-libpq-certs (init)
      name    |       created_at
   -----------+------------------------
    2dcdrtest | <date and time when step 2 was done>
   (1 row)
   ```

   If `2dcdrtest` is returned and has a creation time consistent with when you did step 2, then replication is working. If it is not returned, then either replication is not working, or the data is not yet replicated. If an older creation date is returned, then it's possible the registry was not deleted after a previous check, in which case delete the registry and repeat this test. If replication is not working, follow the 2DCDR troubleshooting steps: Troubleshooting a two data center deployment.
4. Return to the Cloud Manager UI, and delete the `2dcdrtest` user registry.

Verifying Portal replication

Follow these steps to verify that the portal database is replicating from the active data center to the warm-standby data center.

1. In the active data center, identify the active db pod:

```
kubectl get pods -n <namespace> | grep "<active site name>-db"
```

Example output:

```
<portal instance name>-<active site name>-db-0                                    2/2     Running     0
22h
```

where *<active site name>* is the **siteName** property from the portal CR on your active data center. The active data center db pods have 2 containers, so show a status of 2/2. A three replica deployment has three db pods and you can pick any one of them for the next step.

2. In the active data center, run the following command:

```
kubectl exec -ti -n <namespace> <active portal db pod> -c db -- bash -ic "dbstatus 2>&1 | grep Last | sed 's/.*committed:
\([0-9]*\).*/\1/'"
```

Example output:

```
355175
```

where *<active portal db pod>* is the pod name that is returned from step 1. The command returns a number.

3. In the warm-standby data center, identify the warm-standby db pod:

```
kubectl get pods -n <namespace> | grep "<warm-standby site name>-db"
```

Example output:

```
<portal instance name>-<warm-standby site name>-db-0                              2/2     Running     0
22h
```

where *<warm-standby site name>* is the **siteName** property from the portal CR on your warm-standby data center. The warm-standby data center db pods have 1 container, so show a status of 1/1. A three replica deployment has three db pods and you can pick any one of them for the next step.

4. In the warm-standby data center, run the following command:

```
kubectl exec -ti -n <namespace> <passive portal db pod> -c db -- bash -ic "dbstatus 2>&1 | grep Last | sed 's/.*committed:
\([0-9]*\).*/\1/'"
```

Example output:

```
355495
```

where *<passive portal db pod>* is the pod name that is returned from step 3. The command returns a number.

5. The numbers that are returned from steps 2 and 4 represent the portal database's last committed transaction number. When the active data center commits a new transaction the number is incremented, this update is replicated to the warm-standby data center. If replication is working the number that is returned from the warm-standby data center should always be close to the number that is returned from the active data center. It is unlikely to be the same number that is returned, unless you are able run step 4 and step 2 at precisely the same time.
6. Repeat steps 2 and 4 several times to confirm that the warm-standby number is keeping up with the active number. If the number returned from the warm-standby data center is far behind, or ahead of the number that is returned from the active, then there is likely to be a replication problem, see Troubleshooting a two data center deployment.

Synchronizing the ingress-ca certificates between sites

The X.509 CA certificate that is used for TLS communication between the API Connect deployments in each data center is referred to as the **ingress-ca** X.509 certificate. This certificate is stored in the **ingress-ca** **secret** in both data centers, and it must be the same in each data center for 2DCDR to function.

The cert-manager has a separate object that is referred to as the **ingress-ca** Kubernetes cert-manager certificate object. Cloud Pak for Integration and OpenShift top-level CR deployments has the **ingress-ca** Kubernetes cert-manager certificate object in both data centers. Kubernetes and OpenShift individual subsystem installations, has the **ingress-ca** Kubernetes cert-manager certificate object in only one of the data centers, the data center that was the original active data center when 2DCDR was first configured.

Ingress-ca on Kubernetes and OpenShift individual subsystem installations

In Kubernetes and OpenShift individual subsystem CR installations, the **ingress-ca** Kubernetes cert-manager certificate object exists on only one of the data centers in a 2DCDR deployment. The **ingress-ca** **secret** object exists in both data centers and must contain the same certificate string:

```
kubectl -n <namespace> get secret ingress-ca -o yaml

apiVersion: v1
data:
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
...
```

To identify the data center that has the **ingress-ca** Kubernetes cert-manager certificate object, run:

```
kubectl -n <namespace> get cert ingress-ca

   NAME         READY   SECRET       AGE
   ingress-ca   True    ingress-ca   30h
```

Note: If the data center that has the **ingress-ca** Kubernetes cert-manager certificate object is in a failed state and is unrecoverable, a new **ingress-ca** Kubernetes cert-manager certificate object can be created on the active data center by using the **ingress-ca** certificate section of the installation YAML file: **helper_files/ingress-issuer-v1.yaml**:

```
apiVersion: cert-manager.io/v1
kind: Certificate
```

```
metadata:
  name: ingress-ca
  labels: {
    app.kubernetes.io/instance: "management",
    app.kubernetes.io/managed-by: "ibm-apiconnect",
    app.kubernetes.io/name: "ingress-ca"
  }
spec:
  secretName: ingress-ca
  commonName: "ingress-ca"
  usages:
  - digital signature
  - key encipherment
  - cert sign
  isCA: true
  duration: 87600h # 10 years
  renewBefore: 720h # 30 days
  privateKey:
    rotationPolicy: Always
  issuerRef:
    name: selfsigning-issuer
    kind: Issuer
```

Synchronizing the `ingress-ca` X.509 certificate across data centers

Follow these steps to extract your **ingress-ca** X.509 certificate from your source data center and prepare it for application on your target data center:

1. Extract the **ingress-ca secret** from your source data center to a file called **new-ca-issuer-secret.yaml**:

   ```
   kubectl get secret ingress-ca -o yaml -n <namespace>  > new-ca-issuer-secret.yaml
   ```

2. Edit the **new-ca-issuer-secret.yaml** file and remove the **creationTimestamp**, **resourceVersion**, **uid**, **namespace**, and **managedFields**. Remove the labels and annotations sections completely. The resulting contents should include the **ingress-ca** X.509 certificate, and the secret name:

   ```
   apiVersion: v1
   data:
     ca.crt: <long cert string>
     tls.crt: <long cert string>
     tls.key: <long cert string>
   kind: Secret
   metadata:
     name: ingress-ca
   type: kubernetes.io/tls
   ```

3. Copy the **new-ca-issuer-secret.yaml** to the target data center.

Follow these steps to apply the extracted **ingress-ca** X.509 certificate on your target data center:

1. To apply the **new-ca-issuer-secret.yaml** file, run:

   ```
   kubectl apply -f new-ca-issuer-secret.yaml -n <namespace>
   ```

2. Regenerate all **ingress-ca** end-entity certificates:

   ```
   kubectl get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-
   manager\.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n
   <namespace>
   ```

   All affected pods should automatically restart. For more information about regenerating certificates, see: Renewing cert-manager controlled certificates.

Troubleshooting a two data center deployment

What to check when a 2DCDR deployment is not starting or not replicating data.

Typical causes of 2DCDR failures are network problems between data centers, TLS certificate mismatches, or environmental problems on one of the data centers.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes **kubectl** command. On OpenShift, use the equivalent **oc** command in its place.

Check that API Connect pods are running

Confirm that all the pods are ready and running:

- To check portal, run:

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

  ```
  NAME                        READY                                 STATUS    VERSION    RECONCILED VERSION   MESSAGE
  AGE
  <portal instance name>      <ready pods>/<total number of pods>   Running   10.0.5.x   10.0.5.x-xxxx        Serving 1
  site   14d
  ```

- To check management, run:

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

  ```
  NAME                        READY                                 STATUS    VERSION    RECONCILED VERSION   AGE
  <management instance name>  <ready pods>/<total number of pods>   Running   10.0.5.x   10.0.5.x-xxxx        14d
  ```

The number of *<ready pods>* must match the *<total number of pods>*. If not all the pods are in ready state, run

```
kubectl get <mgmt or ptl> -n <namespace> -o yaml
```

and check the `status.conditions` output, for example:

```
  conditions:
  - lastTransitionTime: "2022-09-27T21:37:21Z",
    message: "Management installation in progress. Not all services are ready, pending services: analytics-proxy, apim, client-
downloads-server, juhu, ldap, ...",
    reason: na
    status: "True"
    type: Pending
```

Also, check which pods are not running with:

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

and review their logs:

```
kubectl logs <pod name> -n <namespace>
```

Note: The warm-standby management subsystem has fewer pods than the active management subsystem. The active and warm-standby portal subsystems have the same number of pods.

Check the multiSiteHA sections of the management and portal CRs

Run `kubectl describe` on your management and portal CRs to confirm that the `multiSiteHA` specification and status sections of the CR are as expected. For example, on the warm-standby management CR you should see:

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

```
...
spec:
  Multi Site HA:
    Mode:  passive
    Replication Endpoint:
      Annotations:
        cert-manager.io/issuer:  ingress-issuer
      Hosts:
        Name:                mgmt-replication.apps.example.com
        Secret Name:         mgmt-replication-server
    Replication Peer FQDN:  mgmt-replication.apps.example.com
    Tls Client:
      Secret Name:  mgmt-replication-client
...
Status
...
  Ha Mode:                                passive
...
  Multi Site HA:
    Remote Site Deployment Name:  management
    Remote Site Name:             passive
    Remote UUID:                  c736b702-b1ab-4fe2-b132-9c9d3b3a3bd3.9f986be1-14fc-4b3a-8eee-094217ce361e
    Replication Peer:             https://mgmt-replication.apps.example.com/
```

Check that the `Hosts` and `Replication Peer` properties are set correctly in the `multiSiteHA` section, and that the hostnames resolve correctly in each data center.

Check the `ingress-ca` X.509 certificates match

Replication fails if both data centers do not have the same X.509 certificate in their `ingress-ca` secrets. Run the following command in both data centers to see the X.509 certificate, and check that the output is the same:

```
openssl x509 -noout -fingerprint -sha256 -in <(kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk
'{print $2}' | base64 -d)
```

If you do not have the `openssl` command available, you can instead run only the `kubectl` part, which produces a larger output:

```
kubectl get secret ingress-ca -n <namespace> -o yaml | grep "^  tls.crt:" | awk '{print $2}' | base64 -d
```

if the outputs are different, follow these steps to synchronize the certificates:

1. If you installed API Connect on Kubernetes or OpenShift using individual subsystem CRs, determine which data center has the `ingress-ca` Kubernetes cert-manager certificate object:

   ```
   kubectl get certificates -n <namespace> | grep ingress-ca
   ```

   this is your source data center.
2. If you installed API Connect on Cloud Pak for Integration, use the current active data center as the source data center, and the warm-standby as the target data center.
3. Extract the `ingress-ca` **secret** from your source data center to a file called `new-ca-issuer-secret.yaml`:

   ```
   kubectl get secret ingress-ca -o yaml -n <namespace>  > new-ca-issuer-secret.yaml
   ```

4. Edit the `new-ca-issuer-secret.yaml` file and remove the `creationTimestamp`, `resourceVersion`, `uid`, `namespace`, and `managedFields`. Remove the labels and annotations sections completely. The resulting contents should include the `ingress-ca` X.509 certificate, and the secret name:

   ```
   apiVersion: v1
   data:
   ```

```
  ca.crt: <long cert string>
  tls.crt: <long cert string>
  tls.key: <long cert string>
kind: Secret
metadata:
  name: ingress-ca
type: kubernetes.io/tls
```

5. Copy the `new-ca-issuer-secret.yaml` to the target data center.

Follow these steps to apply the extracted `ingress-ca` X.509 certificate on your target data center:

1. To apply the `new-ca-issuer-secret.yaml` file, run:

   ```
   kubectl apply -f new-ca-issuer-secret.yaml -n <namespace>
   ```

2. Regenerate all `ingress-ca` end-entity certificates:

   ```
   kubectl get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-
   manager\.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n
   <namespace>
   ```

   All affected pods should automatically restart. For more information about regenerating certificates, see: Renewing cert-manager controlled certificates.

Check the operator and tunnel pod logs

The API Connect operator pod manages the 2DCDR deployment. The tunnel pods manage the communication between data centers.

Check the logs of the API Connect operator pod and search for the text "multi" to see any errors that are related to 2DCDR. For example:

```
kubectl logs <ibm-apiconnect operator pod> -n <operator namespace> | grep -i multi
```

The *<ibm-apiconnect operator pod>* has `ibm-apiconnect` in its name, and might be in a different namespace to your API Connect operand pods.

Check the logs of your API Connect `tunnel` pods. These pods always have `tunnel` in their name. For example:

```
kubectl logs portal-tunnel-0 -n <namespace> --since=10m
```

```
...
[ ws-tunnel stderr]   400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.786Z    INFO    tls     incoming
request {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr]   400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.786Z    INFO    tls     connect to
upstream        {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr]   400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 2022-12-02T12:27:33.817Z    INFO    tls     closing
connection      {"remote-addr": "10.254.20.144:55042", "uri": "/portal-active-db-0/3060"}
[ ws-tunnel stderr]   400 0eec87:fb8309:21aa2d 2022-12-02 12:27:33: 10.254.20.144 - - [02/Dec/2022:12:27:33 +0000] "GET
/portal-active-db-0/3060 HTTP/1.1" 101 0
```

Note: It is normal for the management tunnel pod to repeatedly log:

```
2022/12/02 12:29:17 http: TLS handshake error from 10.254.16.1:44812: EOF
```

This message can be filtered out with `grep -v`:

```
kubectl logs management-tunnel-574bdcd865-48zh6 -n <namespace> | grep -v "http: TLS handshake error from"
```

Renewing TLS certificates

Renew TLS certificates in your API Connect deployment.

A default API Connect deployment uses cert-manager to create *issuers*, *CA certificates*, *server certificates*, and *client certificates*. Cert-manager monitors all the certificates that it creates, and renews them before they expire.
Note: If you are not using cert-manager, or if you customized some or all or your API Connect certificates, then you are responsible for monitoring and renewing those certificates.
Certain API Connect configuration and maintenance operations can require that some or all TLS certificates that are used by API Connect are renewed before their expiry. The process of updating API Connect TLS certificates is called "certificate renewal". The process to renew certificates involves the following steps:

1. Manually deleting Kubernetes secrets that contain the TLS certificates that need renewal.
2. Cert-manager detecting the deletion of these secrets, and automatically creating new secrets that contain newly generated x509 certificates.

Steps steps for identifying and renewing API Connect certificates are described in the following topics:

- **Renewing cert-manager controlled certificates**
  Use cert-manager to renew the issuers, CA certificates, and derived certificates that it manages for your API Connect deployment.
- **Renewing custom certificates**
  If you did not use cert-manager to generate certificates for your API Connect deployment, you must renew them manually.
- **List of Issuers, CA certificates, and secrets**
  A summary of certificate issuers, with the CA issued by each, and the corresponding secret used to sign certificates in an API Connect deployment.
- **List of ingress certificates**
  A summary of certificates that are used for communications between subsystems and clients in an API Connect deployment.
- **List of intra-subsystem certificates**
  A summary of certificates used for communications within subsystems in an API Connect deployment.
- **V2018 upgrade: List of certificates to update manually**
  If you upgraded to API Connect V10 from V2018, some external certificates are carried over from the V2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually.

- **Renewing certificates in a two data center deployment on Kubernetes and OpenShift**
  Renewing TLS certificates in an API Connect two data center disaster recovery deployment requires you to copy some updated information between data centers while ensuring that the shared information is not overwritten during the remaining updates.

Related information

- API Connect TLS certificates reference page.

Renewing cert-manager controlled certificates

Use cert-manager to renew the issuers, CA certificates, and derived certificates that it manages for your API Connect deployment.

Cert-manager monitors and automatically renews all API Connect certificates before they expire. However, some API Connect maintenance operations require the manual renewal of certificates. The topics in this section describe how to manually trigger cert-manager to renew the certificates that it manages.

Key points to understand:

- If you replaced any cert-manager certificates with custom certificates, you must update these certificates manually before they expire. Cert-manager does not monitor custom certificates.
- Some API Connect pods must be restarted when certain API Connect certificates are renewed. See Pods that require restart after certificate renewal.
- When you renew a CA certificate, you must renew all the end-entity certificates that the CA signs. See API Connect TLS certificates reference.

Note: If your API Connect deployment was originally installed at v10.0.1.2 or earlier, then the certificate might be missing the `duration` and `renewBefore` properties. Before you renew any API Connect connect certificates, verify that your certificates have these properties, and if not then add them:

1. Run the following command to list of all the certificate names, with their `duration` and `renewBefore` properties:

   ```
   kubectl -n <namespace> get certificate -o custom-
   columns=NAME:metadata.name,DURATION:spec.duration,RENEWBEFORE:spec.renewBefore
   ```

   Example output where `management-ca` is missing the `duration` and `renewBefore` properties:

   ```
   NAME                                    DURATION     RENEWBEFORE
   analytics-ai-endpoint                   17520h0m0s   720h0m0s
   ...
   management-ca                           <none>       <none>
   ```

2. For each certificate output from step 1 that shows `<none>` for `duration` and `renewBefore`, edit the certificate and add these properties:

   ```
   kubectl -n <namespace> edit certificate <certificate name>
   ```

   Add the `duration` and `renewBefore` properties under the `spec` property as shown in the following example:

   ```
   ...
   spec:
   ...
     duration: <duration>
     renewBefore: 720h # 30 days
   ...
   ```

   where *<duration>* is:
   - `87600h` for CA certificates.
   - `17520h` for all other certificates.

- **Renewing the management-ca with the ManagementSecretRotation CR**
  Use the `ManagementSecretRotation` CR to renew the `management-ca` certificate and all end-entity certificates that `management-ca` signs.
- **Renewing the portal-ca with the PortalSecretRotation CR**
  Use the `PortalSecretRotation` CR to renew the `portal-ca` and all end-entity certificates that the `portal-ca` certificate signs.
- **Renewing the analytics CA**
  Renew the analytics CA certificate, and all end-entity certificates that the analytics CA signs.
- **Renewing the ingress-ca**
  Renew the ingress CA certificate, and all the end-entity certificates that the ingress CA signs.
- **Renewing end-entity certificates**
  Renew an API Connect end-entity certificate.
- **Monitoring cert-manager certificate renewal**
  How to monitor when your certificates are nearing their expiry, confirm that cert-manager is renewing them, and verify that dependent certificates are also renewed and related pods are restarted.

Renewing the `management-ca` with the ManagementSecretRotation CR

Use the `ManagementSecretRotation` CR to renew the `management-ca` certificate and all end-entity certificates that `management-ca` signs.

About this task

Applying the `ManagementSecretRotation` CR (Custom Resource) is the recommended method for renewing the `management-ca` and all its end-entity certificates. The alternative method is to manually renew the `management-ca` certificate and then each of its end-entity certificates, which you can identify from the certificates table Management certificates.

Restriction: The **ManagementSecretRotation** CR is for use with a single data center deployment. Do not attempt to use it with a two data center disaster recovery deployment. See Renewing certificates in a two data center deployment on Kubernetes and OpenShift.

Procedure

1. Create a file called management_cert_rotate.yaml and paste in the following contents:

```
apiVersion: management.apiconnect.ibm.com/v1beta1
kind: ManagementSecretRotation
metadata:
  name: mgmt-rotate-issuer
spec:
  managementCluster: <management CR name>
  rotateCertificates:
    certificates:
    - <management CR name>-ca
```

   where *<management CR name>* is the name of your **ManagementCluster** CR. You can identify this name with:

```
kubectl get ManagementCluster -n <management namespace>
```

2. Apply the CR by running the following command:

```
kubectl create -f management_cert_rotate.yaml -n <management namespace>
```

   Applying the CR updates the **management-ca**, and all end-entity certificates that **management-ca** signs. To view a list of the renewed certificates, run the following command:

```
kubectl describe ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

   The **status** block in the command output shows the renewed certificates:

```
Status:
...
  Phase:                    Completed
  Rotated Certs:
    def-management-ca
    def-management-client
    def-management-db-client-postgres
    def-management-natscluster-mgmt
    def-management-site1-postgres
    def-management-db-client-apicuser
    def-management-server
  State:  1/1
...
```

3. When certificate rotation is successfully completed, delete the **ManagementSecretRotation** CR.
   Confirm that certificate rotation is finished with the command:

```
kubectl get ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

   The output should show **Completed**:

```
NAME                 READY   STATUS      AGE
mgmt-rotate-issuer   1/1     Completed   6m52s
```

   Then delete the CR:

```
kubectl delete ManagementSecretRotation mgmt-rotate-issuer -n <management namespace>
```

Renewing the portal-ca with the PortalSecretRotation CR

Use the **PortalSecretRotation** CR to renew the **portal-ca** and all end-entity certificates that the **portal-ca** certificate signs.

About this task

Applying the **PortalSecretRotation** CR (Custom Resource) is the recommended method for renewing the **portal-ca** certificate and all its end-entity certificates. The alternative method is to manually renew the **portal-ca** certificate and then each of its end-entity certificates, which you can identify from the certificates table Portal certificates.

Renewing the **portal-ca** certificate causes downtime because all the **portal-db** pods need to restart the MySQL process to allow for the certificate to update. The pods do not restart, but they go non-ready and then ready again, which takes a few minutes.

Restriction: The **PortalSecretRotation** CR is for use with a single data center deployment. Do not attempt to use it with a two data center disaster recovery deployment. See Renewing certificates in a two data center deployment on Kubernetes and OpenShift.

Procedure

1. Create a file called portal-secret-cr.yaml and paste in the following contents:

```
apiVersion: portal.apiconnect.ibm.com/v1beta1
kind: PortalSecretRotation
metadata:
  name: portal-rotate-secret
spec:
```

```
  portalCluster: <portal CR name>
  # List of certificates you want to rotate (Listing an issuer will rotate any certs issued by the issuer e.g listing just
portal-ca will rotate portal-ca, portal-client and portal-server)
  rotateCertificates:
    certificates:
    - <portal CR name>-ca
  rotateEncryptionSecret:
    # Set to true if you want to rotate the encryption secret.
    rotate: true
    # Optional value to set the encryption secret to, if supplied secret exists in the env. If not supplied the rotated
secret will be random.
    encryptionSecret: new-encyption-secret
```

where *<portal CR name>* is the name of your **PortalCluster** CR. You can identify this name with:

```
kubectl get PortalCluster -n <portal namespace>
```

If you set **rotate: true** in the **rotateEncryptionSecret** section, the portal-encryption-secret is updated and the following changes are made:

- Change the root MySQL password.
- Generate a new, site-specific MySQL password for each site.
- Reencrypt all client_ids and client_secrets in the portal database.

2. Apply the CR by running the following command:

```
kubectl create -f portal-secret-cr.yaml -n <portal namespace>
```

Applying the CR updates the **<portal CR name>-ca**, along with all end-entity certificates that it signs. To view a list of the updated certificates, run the following command:

```
kubectl describe PortalSecretRotation portal-rotate-secret -n <portal namespace>
```

The **status** block in the command output shows the renewed certificates:

```
Status:
  ...
  Phase:                    Completed
  Rotated Certs:
    def-portal-ca
    def-portal-server
    def-portal-client
  State:  Completed Portal Secret Rotation.
...
```

3. When the certificate rotation is finished, delete the **PortalSecretRotation** CR.
Confirm that certificate rotation is finished with the command:

```
kubectl get PortalSecretRotation portal-rotate-issuer -n <portal namespace>
```

The output should show **Completed**:

```
NAME                   STATUS      MESSAGE                              AGE
portal-rotate-secret   Completed   Completed Portal Secret Rotation.    7m50s
```

Then delete the CR:

```
kubectl delete PortalSecretRotation portal-rotate-issuer -n <portal namespace>
```

Renewing the analytics CA

Renew the analytics CA certificate, and all end-entity certificates that the analytics CA signs.

About this task

The pods that comprise the analytics subsystem communicate with each other using certificates that are signed by the analytics CA certificate. If you renew the analytics CA certificate, you must also renew the analytics end-entity certificates that the analytics CA signs.

For more information about API Connect certificates, see API Connect TLS certificates.

Procedure

1. Run the following command to renew the analytics CA certificate:

```
kubectl -n <analytics namespace> get certificate <analytics CA name> -o=jsonpath='{.spec.secretName}' | xargs kubectl -n
<analytics namespace> delete secret
```

where *<analytics CA name>* is the name of the analytics CA certificate. On Kubernetes and OpenShift individual subsystem installations this name is **analytics-ca**. On Cloud Pak for Integration and OpenShift top-level CR installations, this name is **<apic instance name>-a7s-ca**.
Verify that a new *<analytics CA name>* **secret** is created:

```
kubectl -n <analytics namespace> get secret
```

Output should show a new *<analytics CA name>* secret:

```
NAME                                                    TYPE                DATA   AGE
...
<analytics CA name>                                     kubernetes.io/tls   4      28s
```

Check that the **AGE** column shows the secret was recently created.

2. Renew the end-entity certificates signed by the analytics CA. Run the following command for each certificate:

   ```
   kubectl get certificate <analytics certificate> -o=jsonpath='{.spec.secretName}' | xargs kubectl delete secret
   ```

   where *<analytics certificate>* is the name of the end entity certificate. The analytics end-entity certificates are:
   - **analytics-client** or **a7s-client**.
   - **analytics-server** or **a7s-server**.

   On Cloud Pak for Integration and OpenShift top-level CR installations, the certificate name is prefixed with ***<apic instance name>-***.
3. Restart the analytics pods so that they use the renewed certificates: Restarting analytics pods.

Renewing the ingress-ca

Renew the ingress CA certificate, and all the end-entity certificates that the ingress CA signs.

About this task

The ingress CA certificate signs all user-facing and inter-subsystem certificates. When you renew the ingress CA, you must renew all the end-entity certificates that the ingress CA signs.
For more information about API Connect certificates, see API Connect TLS certificates.

Procedure

1. Run the following command to renew the ingress CA certificate:

   ```
   kubectl -n <management namespace> get certificate <ingress CA name> -o=jsonpath='{.spec.secretName}' | xargs kubectl -n <management namespace> delete secret
   ```

   where *<ingress CA name>* is the name of the ingress CA certificate. On Kubernetes and OpenShift individual subsystem installations this name is **ingress-ca**. On Cloud Pak for Integration and OpenShift top-level CR installations, this name is ***<apic instance name>*-ingress-ca**.
2. Run the following command to renew all the end-entity certificates that the ingress CA signs:

   ```
   kubectl get secrets -n <management namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager\.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <management namespace>
   ```

3. If you have other subsystems that are in different namespaces from the management subsystem, then follow the steps in Copying renewed ingress-ca to subsystems in different namespaces.
4. Verify analytics in the developer portal.
   Due to a known issue, when the **ingress-ca** is renewed, it is possible that Analytics in the Developer Portal might stop working. If this happens, complete the following steps to ensure that certificate changes take effect:
   a. Log in to the Cloud Manager user interface.
   b. In the navigation list, click Topology.
   c. Edit the Analytics service.
   d. On the Analytics page, edit the Summary field to force a change; for example, by adding a space to the end of a sentence.
   e. Click Save.

Copying renewed ingress-ca to subsystems in different namespaces

When the ingress CA certificate on the management subsystem is renewed, you must copy the renewed certificate to any subsystems that are in a different namespace.

Before you begin

Complete the steps in Renewing the ingress-ca to renew the ingress CA in your management subsystem namespace.
Note: If all your API Connect subsystems are in the same namespace, you do not need to follow this procedure.

About this task

The ingress CA signs all the inter-subsystem communication certificates that are used between API Connect subsystems. If your subsystems are installed in different namespaces, then the ingress CA must be manually synchronized across all those namespaces.

These steps must be repeated for all namespaces where you have subsystems that are registered with your management subsystem.

For more information about API Connect certificates, see API Connect TLS certificates.

Procedure

1. Export the management namespace ingress CA secret to a YAML file called ingress-ca.yaml:

   ```
   kubectl -n <management namespace> get secret ingress-ca -o yaml > ingress-ca.yaml
   ```

2. Edit **ingress-ca.yaml** and remove all **metadata** fields except for the secret name. The resulting file contents look like this:

```
apiVersion: v1
data:
  ca.crt: ...
  tls.crt: ...
  tls.key ...
kind: Secret
metadata:
  name: ingress-ca
type: kubernetes.io/tls
```

3. Apply the updated the `ingress-ca.yaml` in the namespace of the other subsystem:

   ```
   kubectl -n <other subsystem namespace> apply -f ingress-ca.yaml
   ```

4. Verify that the ingress CA is now identical in both namespaces.
   a. Extract your `ingress-ca` secret to a file in your management namespace:

      ```
      kubectl -n <management namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print $2}'
      | base64 -d > ingress.pem.mgmt
      ```

   b. Extract your `ingress-ca` secret to a file in your other subsystems' namespace:

      ```
      kubectl -n <other subsystem namespace> get secrets ingress-ca -o yaml | grep tls.crt | grep -v 'f:tls' | awk '{print
      $2}' | base64 -d > ingress.pem.other
      ```

   c. Verify that the files that are created for each namespace are identical:

      ```
      diff ingress.pem.mgmt ingress.pem.mgmt
      ```

5. Run the following command to renew all the `ingress-ca` end-entity certificates in the other subsystems' namespace:

   ```
   kubectl get secrets -n <other subsystem namespace> -o custom-
   columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager\.io/issuer-name' --no-headers=true | grep ingress-
   issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <other subsystem namespace>
   ```

Renewing end-entity certificates

Renew an API Connect end-entity certificate.

About this task

To renew an end-entity certificate, delete the secret that corresponds to the certificate. Cert-manager detects the deletion of the secret and re-creates it with a new x509 certificate.

For more information about API Connect certificates, see API Connect TLS certificates.

Procedure

1. Get a list of all certificates in your API Connect namespace:

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

   The output lists all certificates with their status, age, and the name of the corresponding secret. In this example, the certificate that is used by the API Manager UI is shown:

   ```
   NAME                                    READY   SECRET                                  AGE
   ...
   apim-endpoint                           True    apim-endpoint                           13d
   ...
   ```

   By default the secret names are identical to the certificate names.

2. Delete the secret for the certificate that you want to renew:

   ```
   kubectl -n <namespace> delete secret <secret name>
   ```

   Cert-manager detects the deletion of the secret and creates a new secret with a newly generated x509 certificate.

Monitoring cert-manager certificate renewal

How to monitor when your certificates are nearing their expiry, confirm that cert-manager is renewing them, and verify that dependent certificates are also renewed and related pods are restarted.

CA certificate renewal

If cert-manager renews a CA certificate, you must renew all the certificates that the CA signs. See API Connect TLS certificates reference.

Monitoring when your certificates are due to expire

Check when your certificates are due to expire with the following command:

```
kubectl -n <namespace> get certificate -o custom-
columns=NAME:metadata.name,DURATION:spec.duration,RenewBEFORE:spec.renewBefore,NotAFTER:status.notAfter
```

Example output:

```
NAME                                DURATION     RenewBEFORE   NotAFTER
analytics-ingestion-client          17520h0m0s   720h0m0s      2025-08-28T08:25:58Z
api-endpoint                        17520h0m0s   720h0m0s      2025-08-28T08:28:40Z
apim-endpoint                       17520h0m0s   720h0m0s      2025-08-28T08:28:21Z
```

The example output shows that your `analytics-ingestion-client` is due to expire at `2025-08-28T08:25:58Z`. The `RenewBEFORE` property shows that cert-manager should renew this certificate 720 hours before it expires.

Monitoring the cert-manager manager logs

Monitor the cert-manager logs to confirm when certificates are renewed. Run the following command to display the cert-manager log:

```
kubectl logs <cert-manager pod name> -n <cert-manager namespace>
```

Example output:

```
I0108 14:23:18.001074       1 controller.go:129] cert-manager/controller/certificates "level"=0 "msg"="syncing item"
"key"="default/test-cert"
I0108 14:23:18.009940       1 issue.go:109] cert-manager/controller/certificates/certificates/issue "level"=0
"msg"="certificate issued" "related_resource_kind"="Secret" "related_resource_name"="test-cert"
"related_resource_namespace"="default"
I0108 14:23:18.018664       1 sync.go:331] cert-manager/controller/certificates/certificates "level"=0 "msg"="certificate
scheduled for renewal" "duration_until_renewal"="-5.018566857s" "related_resource_kind"="Secret" "related_resource_name"="test-
cert" "related_resource_namespace"="default"
I0108 14:23:18.019070       1 controller.go:135] cert-manager/controller/certificates "level"=0 "msg"="finished processing work
item" "key"="default/test-cert"
```

The 4 statements (shown in the example) indicate when a certificate is about to be renewed, and when the renewal is complete a few seconds later.
Tip: Set up a logs-based alerting mechanism that informs you when cert-manager renews any certificate.

Pods that require restart after certificate renewal

Some API Connect pods must be restarted when certificates that are used by the pod are renewed. The certificates that require pods to be restarted after renewal are:

- `analytics-ingestion-client` or `a7s-ing-client`. Restart `apim`, `taskmanager`, and `analytics-proxy` pods on the management subsystem.
- `gateway-client-client` or `gw-dr-client`. Restart the `apim` and `taskmanager` pods on the management subsystem.
- `analytics-ai-endpoint` or `a7s-ai-endpoint`. Restart the `mtls-gw` pod on the analytics subsystem.

Renewing custom certificates

If you did not use cert-manager to generate certificates for your API Connect deployment, you must renew them manually.

Procedure

1. Renew the certificates that you created:
   Use one of the following methods to renew the certificates that you created:
   - Update the secret that contains the certificate with your new certificate.
   - Delete and then re-create the secret with a new certificate.
   - Create a new secret (with a new name) that contains your new certificate. Update the appropriate API Connect CR to reference the new secret name.
2. Some pods require a restart after you update certain certificates. Refer to API Connect certificate reference to check whether any of the certificates you updated require pod restarts.

List of Issuers, CA certificates, and secrets

A summary of certificate issuers, with the CA issued by each, and the corresponding secret used to sign certificates in an API Connect deployment.

Table 1 presents a list of the certificate issuers used in an API Connect deployment, with the CA issued by each, and the corresponding secret used to sign certificates. The `selfsigning-issuer` is used only for creating CAs, and does not have a corresponding certificate or secret of its own. If you have customized certificates, the certificates generated by the issuers in Table 1 use the custom-certs-external.yaml template.

Table 1. Issuers, CAs, and secrets

Issuer	CA certificate	Secret
`selfsigning-issuer`	N/A	N/A
`ingress-issuer`	`ingress-ca` issued by `selfsigning-issuer`	`ingress-ca` Attention: If you change the `ingress-ca` secret in a two data center deployment, you must change it to the same value in both data centers.
`analytics-ca`	`analytics-ca` issued by `selfsigning-issuer`	`analytics-ca`
`management-ca`	`management-ca` issued by `selfsigning-issuer`	`management-ca`

Issuer	CA certificate	Secret
`portal-ca`	`portal-ca` issued by `selfsigning-issuer`	`portal-ca`

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `managment-ca` is called `<apic instance name>-mgmt-ca`.

List of ingress certificates

A summary of certificates that are used for communications between subsystems and clients in an API Connect deployment.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `managment-ca` is called `<apic instance name>-mgmt-ca`.

User-facing certificates

Table 1 presents a list of user-facing certificates. The corresponding secrets use the same names. If you customized your certificates, the certificates in Table 1 use the custom-certs-external.yaml template.

Table 1. User-facing certificates

Certificate name	Issuer	Description
`api-endpoint or mgmt-platform-api`	`ingress-issuer`	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
`consumer-endpoint or mgmt-consumer-api`	`ingress-issuer`	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
`apim-endpoint or mgmt-api-manager`	`ingress-issuer`	API Manager UI server certificate.
`cm-endpoint or mgmt-admin`	`ingress-issuer`	Cloud Manager UI server certificate.
`portal-web`	`ingress-issuer`	The server certificate used on the `portalUIEndpoint`. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site. The default `portal-web` certificate is issued by the `ingress-issuer`, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate. This certificate is used by the `portal-web` ingress (`portal-web` route on OpenShift).
`hub-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `hub-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`turnstile-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `turnstile-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.

Inter-subsystem certificates

Table 2 lists all of the common subsystem communication certificates. The corresponding secrets use the same names. If you have customized your certificates, the certificates in Table 2 use the custom-certs-external.yaml template.

Table 2. Inter-subsystem certificates

Certificate name	Issuer	Description
`-ingress-ca`	`selfsigning-issuer`	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem `apim` pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the `portal-www` pods for the update to take effect.

Certificate name	Issuer	Description

`analytics-ingestion-client or a7s-ing-client`	`ingress-issuer`	Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:
		• The same CA as the server certificate `analytics-ai-endpoint` on the analytics subsystem.
		• The common name must match the `spec.clientSubjectDN` in the analytics CR:
		`kubectl describe cert analytics-admin-client`
		`Spec:`
		`  Common Name:  a7s-ing-client`
		`kubectl get a7s -o yaml`
		`  spec:`
		`    clientSubjectDN: CN=a7s-ing-client`
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=a7s-ingestion-client`, or they could both be `CN=a7s-ingestion-client, O=cert-manager`, but they must be identical.
		To update this certificate, use kubectl. Restart the `apim`, `taskmanager`, and `analytics-proxy` pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.

`portal-admin-client or ptl-adm-client`	`ingress-issuer`	Client certificate used for communication with the portal subsystem on the `portalAdminEndpoint`. This certificate must have:
		• The same CA as the server certificate `portal-admin or ptl-director` on the portal subsystem.
		• The common name must match the `spec.adminClientSubjectDN` in the portal CR:
		`kubectl describe cert portal-admin-client`
		`Spec:`
		`  Common Name:  ptl-adm-client`
		`kubectl get ptl -o yaml`
		`  spec:`
		`    adminClientSubjectDN: CN=ptl-adm-client`
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=portal-admin-client`, or they could both be `CN=ptl-adm-client, O=cert-manager`, but they must be identical.

`gateway-client-client or gw-dr-client`	`ingress-issuer`	Client certificate for communications with the gateway service. Restart the `apim`, and `taskmanager` pods after update. This certificate must have the same CA as the `gwv6-manager-endpoint or gw-gateway-manager` certificate on the gateway.
`api-endpoint or mgmt-platform-api`	`ingress-issuer`	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
`consumer-endpoint or mgmt-consumer-api`	`ingress-issuer`	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
`apim-endpoint or mgmt-api-manager`	`ingress-issuer`	API Manager UI server certificate.
`cm-endpoint or mgmt-admin`	`ingress-issuer`	Cloud Manager UI server certificate.
`mgmt-replication-client`	`ingress-issuer`	2DCDR deployments only. Client certificate used in the warm-standby data center's `<remote-sitename>-postgres` pod to connect to the active data center.
`mgmt-replication-server`	`ingress-issuer`	2DCDR deployments only. Server certificate used in the active data center's `<management_CR>-tunnel` pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the `spec.multiSiteHA.replicationEndpoint` defined in the management CR.
`hub-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `hub-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`turnstile-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `turnstile-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`analytics-ai-endpoint or a7s-ai-endpoint`	`ingress-issuer`	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the `analytics-ai-endpoint` certificate. If this certificate is updated, restart the `mtls-gw` pod for the update to take effect.
`portal-admin or ptl-director`	`ingress-issuer`	Server certificate used on the `portalAdminEndpoint`. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the `portal-admin-client` certificate. This certificate must have the same CA as the `portal-admin-client` certificate on the management subsystem. This certificate is used by the `portal-nginx` pod.

Certificate name	Issuer	Description
portal-web	ingress-issuer	The server certificate used on the portalUIEndpoint. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site. The default portal-web certificate is issued by the ingress-issuer, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate. This certificate is used by the portal-web ingress (portal-web route on OpenShift).
ptl-replication-client	ingress-issuer	2DCDR deployments only. Client certificate that is used in the warm-standby data center's <remote-sitename>-www and <remote-sitename>-db pods to connect to the active data center. This certificate is used by all the portal pods related to 2DCDR replication: portal-db-remote, portal-www-remote, and portal-tunnel.
ptl-replication-server	ingress-issuer	2DCDR deployments only. Server certificate used in the active data center's <portal_CR>-tunnel pod, the counterpart to the ptl-replication-client certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the spec.multiSiteHA.replicationEndpoint defined in the portal CR. This certificate is used by all the portal pods related to 2DCDR replication: portal-db-remote, portal-www-remote, and portal-tunnel.
gateway-peering or gw-peer	ingress-issuer	This certificate secures the communication between gateways in your gateway cluster.
gwv6-endpoint or gw-gateway	ingress-issuer	Legacy certificate that is not used.
gwv6-manager-endpoint or gw-gateway-manager	ingress-issuer	The gatewayManager endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the gateway-client-client certificate on management subsystem
event-gateway-management-client	ingress-issuer	The event-gateway-management-client certificate exists only on Cloud Pak for Integration deployments.

List of intra-subsystem certificates

A summary of certificates used for communications within subsystems in an API Connect deployment.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the APIConnectCluster instance name. For example, the certificate managment-ca is called *<apic instance name>*-mgmt-ca.
Table 1 presents a list of certificates used for communications between pods in the same subsystem. The certificates are managed by cert-manager. For details of all the API Connect certificates, see API Connect certificates.

Table 1. Intra-subsystem certificates

Certificate name	Issuer	Description
management-ca or mgmt-ca	selfsigning-issuer	The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using in-cluster communication. See In-cluster service communication between subsystems
management-client or mgmt-client	management-ca	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
management-server or mgmt-server	management-ca	Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: *.<namespace> *.<namespace>.svc *.<instance name>-server.<namespace>.svc <instance name>-server
db-client-apicuser	management-ca	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only db-client-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
db-client-postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only db-client-primaryuser	management-ca	Intra-subsystem certificate for the management database subsystem.
postgres	management-ca	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only postgres-pgbouncer	management-ca	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only postgres-operator	management-ca	Intra-subsystem certificate for the management database subsystem.

Certificate name	Issuer	Description
`natscluster-mgmt`	`management-ca`	Intra-subsystem certificate for the `nats` pods.
v10.0.6 only `db-client-replicator`	`management-ca`	2DCDR deployments only. Client certificate used by the `<management_CR>-tunnel` pod to connect to the other data center's `<management_CR>-tunnel` pod.
`analytics-ca or a7s-ca`	`selfsigning-issuer`	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.
`analytics-client or a7s-client`	`analytics-ca`	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`analytics-server or a7s-server`	`analytics-ca`	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`<instance name>-storage`
		
		If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`portal-ca or ptl-ca`	`selfsigning-issuer`	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.
`portal-client or ptl-client`	`portal-ca`	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.

`portal-server or ptl-server`	`portal-ca`	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
		
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`*.<instance name>-<site name>-db-all.<namespace>.svc`
		`*.<instance name>-<site name>-www-all.<namespace>.svc`
		`*.<instance name>-<site name>-db-all.<namespace>.svc.cluster.local`
		`*.<instance name>-<site name>-www-all.<namespace>.svc.cluster.local`
		`*.<namespace>.svc.cluster.local`
		`<instance name>-db`
		`#<remote portal CR name>-db # For 2DCDR only.`
		
		<instance name> and *<remote portal CR name>* are truncated if more than 15 characters.
		This certificate is used by all portal pods.

V2018 upgrade: List of certificates to update manually

If you upgraded to API Connect V10 from V2018, some external certificates are carried over from the V2018 deployment and are not managed by cert-manager. When you update any of the retained certificates, you must restart the affected pods manually.

Ingress (front-end) certificates

Table 1 presents a list of the ingress certificates that were carried forward when you upgraded from V2018, and the pods that you must restart when you update each certificate.

If multiple Portal web ingress endpoints have been configured, then the certificates for those additional endpoints must be updated manually as well.

Table 1. V2018 ingress certificates and affected pods

Certificate and Secret	Pods
`client-tls-`***`xxxxxxxxxx`***	• `analytics-mtls-gw-`***`hashed-suffix`***
`ingestion-tls-`***`xxxxxxx`***	• `analytics-mtls-gw-`***`hashed-suffix`***
`platform-api-tls-`***`xxxxxxx`***	N/A
`api-manager-tls-`***`xxxxxxxx`***	N/A
`cloud-manager-tls-`***`xxxxxx`***	N/A
`consumer-api-tls-`***`xxxxxxx`***	N/A
`v5GwName``-datapower-admin-credentials`	N/A
`v5GwName``-apic-gateway-peering-tls`	N/A
`v5GwName``-apic-gateway-service-tls`	N/A
`v6GwName``-datapower-admin-credentials`	N/A
`v6GwName``-apic-gateway-peering-tls`	N/A

Certificate and Secret	Pods
`v6GwName-apic-gateway-service-tls`	N/A
`portal-uuid-admin`	• `portal-nginx`
`www-tls-xxxxxxxxxxxxx`	N/A

Subsystem communication certificates

Table 2 presents a list of the subsystem communication certificates that were carried forward when you upgraded from V2018, and the pods that you must restart when you update each certificate.

Table 2. V2018 subsystem communication certificates and affected pods

Usage	Certificate and Secret	Pods
client	`managementUpgradeName-analytics-ingestion-client`	• `management-apim` • `management-taskmanager` • `gateway` (via webhook, no restart needed)
client	`managementUpgradeName-portal-admin`	• `management-apim` • `management-taskmanager` • `management-portal-proxy`
client	`gw-apic-gateway-service-tls`	• `gwv6` • `gwv5`
clientserver	`gw-apic-gateway-peering-tls`	• `gwv6` (redis) • `gwv5` (redis)

Database encryption key

The database encryption key is carried forward from V2018, and uses the name: `managementUpgradeName-encryption-secret`. You can update the value of this secret as explained in Changing the database encryption key.

Renewing certificates in a two data center deployment on Kubernetes and OpenShift

Renewing TLS certificates in an API Connect two data center disaster recovery deployment requires you to copy some updated information between data centers while ensuring that the shared information is not overwritten during the remaining updates.

About this task

Begin the process by updating certificates as needed on data center 1 (DC1). Then,copy the `ingress-ca` from DC1 to data center 2 (DC2) before updating certificates on DC2. Finally, you will copy encryption keys for the Management and Portal subsystem from DC1 to DC2.

Procedure

Complete the following steps in the sequence shown to ensure that you do not overwrite information that was copied from DC1 while updating DC2.

1. DC1: Renew TLS certificates as explained in Renewing TLS certificates.
2. DC2: Complete the following steps to copy the `ingress-ca` from DC1:
   a. On DC1, export the `ingress-ca` secret to a YAML file by running the following command:

      ```
      kubectl -n namespace get secret ingress-ca -o yaml > ingress-ca.yaml
      ```

   b. Still on DC1, edit the `ingress-ca.yaml` file to remove all labels, annotations, `creationTimestamp`, `resourceVersion`, `uid`, and `selfLink`.
   c. Copy the `ingress-ca.yaml` to DC2.
   d. On DC2, apply the `ingress-ca.yaml` file by running the following command:

      ```
      kubectl -n namespace apply -f ingress-ca.yaml
      ```

   e. Validate that the `ingress-ca` on DC2 matches the `ingress-ca` on DC1:
      i. On DC1, run the following command to save the original `ingress-ca` as a PEM file:

         ```
         kubectl -n namespace get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc1
         ```

      ii. On DC2, run the following command to save the copied `ingress-ca` as a PEM file::

         ```
         kubectl -n namespace get secrets ingress-ca -o yaml | grep tls.crt | awk '{print $2}' | base64 -d > /tmp/ingress.pem.dc2
         ```

      iii. Copy /tmp/ingress.pem.dc1 from DC1 and store it in the /tmp folder on DC2.
      iv. On DC2, run the following command to compare the two files and verify that there are no differences:

         ```
         diff /tmp/ingress.pem.dc1 /tmp/ingress.pem.dc2
         ```

3. DC2: Update all external certificates that are based on the `ingress-ca` (see List of ingress certificates).
4. DC2: Update all of the internal certificates (see List of intra-subsystem certificates).
   At this point, DC1 and DC2 have the same `ingress-ca` secrets. Next, copy the encryption keys for the Management and Portal subsystems from DC1 to DC2.

5. Copy the encryption key for the Management subsystem to DC2 by completing the following steps:
   a. On DC1, copy the management encryption key:
      i. Get the name of the management encryption key by running the following command:

      ```
      $ kubectl -n namespace get mgmt -o yaml|grep enc
      ```

      The response looks like the following example:

      ```
      encryptionSecret: dallas-enc-key
      ```

      Make a note of the secret name for the next step. In the example, the key's name is **dallas-enc-key**.

      ii. Retrieve the secret from the encryption key by running the following command (substitute in the name of the key from the previous step):

      ```
      kubectl get secret dallas-enc-key -o yaml
      ```

      The output looks like the following example:

      ```
      apiVersion: v1
      data:
        encryption_secret.bin:
      VKBNFj7sAOizxvE1H6i+9P31oJHvWWsO+x**********************************EGe/K+x6b3D7FEWGgoyGlWBUJKB4+T21My2iR5rBTov
      pyLiY5g*****************************************tSiRcQKegMPNBPgL829SVBCxuv3I=
      kind: Secret
      metadata:
        creationTimestamp: "2020-08-28T12:53:41Z"
        labels:
          app.kubernetes.io/instance: m1
          app.kubernetes.io/managed-by: ibm-apiconnect
          app.kubernetes.io/name: dallas-enc-key
        name: dallas-enc-key
        namespace: default
        resourceVersion: "43039"
        selfLink: /api/v1/namespaces/default/secrets/m1-enc-key
        uid: 46c92395-9cc2-4421-b2c4-48e472c0cbb1
      type: Opaque
      ```

      iii. Copy the output and save it in a YAML file; for example dc1-enc-key.yaml.
      iv. Edit the file and make the following changes:
      - Delete the contents of the `metadata:` section (retain the section).
      - In the `metadata:` section, add in a `name:` attribute, with `dc1-enc-key` as the value.

      The following example shows the updated file:

      ```
      apiVersion: v1
      data:
        encryption_secret.bin:
      VKBNFj7sAOizxvE1H6i+9P31oJHvWWsO+x**********************************EGe/K+x6b3D7FEWGgoyGlWBUJKB4+T21My2iR5rBTov
      pyLiY5g*****************************************tSiRcQKegMPNBPgL829SVBCxuv3I=
      kind: Secret
      metadata:
        name: dc1-enc-key
      type: Opaque
      ```

      v. Copy the updated file (dc1-enc-key.yaml) to the /tmp folder on DC2.
   b. On DC2, run the following command to create a secret from the encryption key file (dc1-enc-key.yaml) that you copied from DC1:

   ```
   $ kubectl  -n namespace create -f /tmp/dc1-enc-key.yaml
   ```

   The response looks like the following example:

   ```
   secret/dc1-enc-key created
   ```

6. Update the encryption key for the Portal subsystem on DC1 and then copy it to DC2 as explained in Changing the secret for portal data and system tools.

Monitoring and health checks

Monitor your deployment and complete health checks on subsystems.

See:

- **Monitoring on Kubernetes**
  Monitor the API Connect subsystems on Kubernetes.
- **Monitoring on OpenShift**
  Monitor your API Connect deployment on OpenShift.
- **Monitoring on Cloud Pak for Integration**
  Monitor your API Connect deployment on Cloud Pak for Integration.
- **Obtaining simple health check data of Developer Portal sites by using a REST API call**
  Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

Monitoring on Kubernetes

Monitor the API Connect subsystems on Kubernetes.

See:

- **Monitoring subsystems on Kubernetes**
  Use `kubectl get` to check the health of the API Connect clusters in your Kubernetes deployment.
- **Monitoring Postgres disk usage**
  Monitor disk usage by the Postgres database in the management subsystem.
- **Monitoring platform health using Kubernetes events**
  Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

Monitoring subsystems on Kubernetes

Use `kubectl get` to check the health of the API Connect clusters in your Kubernetes deployment.

The APIConnect operator controls the life-cycle of management subsystem microservices. When you install, you create the following custom resources (CRs) in order to install API Connect and all subsystems:

- `ManagementCluster` or `mgmt`
- `AnalyticsCluster` or `a7s`
- `PortalCluster` or `ptl`
- `GatewayCluster` or `gw`
  Every CR has a `status` section. The APIConnect operator regularly updates this section with the current status of the subsystem.

For example, use the following command to obtain the status of `ManagementCluster`:

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

Example:

```
kubectl get mgmt
NAME    READY    STATUS     VERSION       RECONCILED VERSION       AGE
m1      16/16    Running    10.0.6.0       10.0.6.0-44    24m
```

Output:

- `NAME`: Name of the Management Subsystem
- `READY`: Current number of management subsystem which are up, followed by the expected number of management subsystem components which should be up. For example: `16/16`.
- `STATUS`: Current status of Management Subsystem
  During fresh install, upgrade or any maintenance operation, such as changing configuration for S3 backups, Management CR status can go to different states depending on the action, but ultimately it should **always** reach `Running` status.

  If the status says `Warning`, then run `kubectl describe mgmt -n <namespace>` and check for more details in the `Status` section. For example, a known issue on v10.0.6 and previous with WAL archiving:

  ```
  Status:
    ...
    Conditions:
      ...
      Message:                             WAL Archiving is not working. Last successful archive time: 2022-07-26
  20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
      Reason:                              WALArchiveNotWorking
      ...
  ```

  Note: If you see the `WAL Archiving is not working` warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.
- `VERSION`: Version chosen in `spec.version` in the management CR.
- `RECONCILED VERSION`: Current management subsystem version. Includes a build number.
- `AGE`: Age of the management subsystem CR.

Monitoring Postgres disk usage

Monitor disk usage by the Postgres database in the management subsystem.

Postgres Database is the core database used in Management Subsystem. It is important to monitor the Postgres disk usage.

The API Connect operator tracks the current disk usage of the Postgres components, and regularly updates the `ManagementCluster` status. When one or more of the Postgres components occupy 60% of the PVC (persistent volume claim) capacity, the API Connect operator changes the status from `Running` to `Warning`.

API Connect operator **brings down** Postgres if the disk allocation reaches **80%**. In this case, you must contact IBM support.

Note that there are various types of storage classes which can be used to deploy Management Subsystem. For example, `local-storage` or `ceph block`.

Management Cluster disk stats

The `ManagementCluster` instance has `.status.postgresDataStats` where the operator prints the current disk usage of Postgres components. For example:

```
kubectl get mgmt -o json | jq .status.postgresDataStats
[
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
```

```
    "pvcCapacity": 51200,
    "pvcName": "",
    "pvcType": "WAL",
    "pvcUsed": 400,
    "pvcUsedPercentage": 8
  },
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
    "pvcCapacity": 184320,
    "pvcName": "",
    "pvcType": "PostgreSQL",
    "pvcUsed": 95,
    "pvcUsedPercentage": 8
  }
]
```

Warning condition is populated at 60% usage

When one of the Postgres components occupy 60% usage, the operator marks the overall status of the **ManagementCluster** to **Warning** with an appropriate warning message. The operator also updates the **postgresDataStats** section with the current data usage of the Postgres components.

Example of warning state:

```
kubectl get mgmt
NAME               READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE
AGE
stv3-management    6/8     Warning   10.0.7.0   10.0.7.0-5363        Some services are not ready - see status condition for
details   3d20h
```

Example of list of warning conditions:

```
kubectl get mgmt -o json
...
status:
  conditions:
  - lastTransitionTime: "2022-06-02T19:38:09Z"
    message: Warning threshold=60%, Current disk usage=63%, Is wal archiving working?=false.
      Database shutdown starts at 80%. Please contact IBM Support immediately.
    reason: wal_disk_usage_more_than_warning_threshold
    status: "True"
    type: Warning
...
```

If you encounter the warning, contact IBM support to fix the root cause.

Error condition is populated at 80% usage

An error condition is populated when the operator brings down Postgres because is using 80% of available disk space. Postgres is brought down to avoid problems, such as disk corruption, that can occur if the disk becomes completely filled.

```
 status:
  conditions:
- lastTransitionTime: "2022-06-03T00:40:54Z"
    message: Error threshold=80%, Current disk usage=82%, Is wal archiving working?=false.
      Database is in shutdown mode and management services are disabled. Please contact
      IBM Support immediately.
    reason: wal_disk_usage_more_than_error_threshold
    status: "True"
    type: Error
```

If you encounter the error, contact IBM support to fix the root cause.

Important:
When **local-storage** is used, it uses the entire disk that is allocated to the worker node. In some cases, Kubernetes itself can face disk pressure before the operator reacts to 80% usage, and will start evicting pods. In these cases, you might need to increase the disk allocated to the worker node, in order to stabilize the worker node, by following Recovering when disks are filled by the management database V10.0.6.0.

Monitoring platform health using Kubernetes events

Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

What are the Kubernetes events?

The CR for every subsystem of API Connect has a status condition section, which is regularly updated by the API Connect operator with the current status. The API Connect operator additionally generates a Kubernetes event whenever there is a change in the CR status.

Events are enabled in API Connect operator for the following resources.

- **APIConnectCluster** CR
- **ManagementCluster** CR
- **GatewayCluster** CR
- **AnalyticsCluster** CR
- **PortalCluster** CR

Two types of events are generated when the CR status changes:

- **`Normal`** - Indicates that the status change is not causing any issues.
- **`Warning`** - Indicates that there a problem that requires your attention.

You can automatically monitor the status condition in your existing alerting system, or by integrating the Kubernetes events into a third-party alerting solution. For more information, see the following blog post: How to use Kubernetes events for effective alerting and monitoring.

Viewing the events

The default retention period of events in Kubernetes is 1 hour (controlled by **`kube-apiserver`**). To view the events, complete the following steps:

1. Run the **`kubectl describe`** against the resource you want to monitor; for example:

   ```
   kubectl describe APIConnectCluster
   ```

2. In the response, scroll to the end to see the events information, which looks like the following example:

   ```
          Name:           10.0.5.3-1
     Reconciled:          10.0.5.3-1
   Events:
     Type    Reason     Age    From                       Message
     ----    ------     ----   ----                       -------
     Normal  Running    10h    ibm-apiconnect-operator    All services ready.
     Normal  Deploying  10h    ibm-apiconnect-operator    Not all services are ready, pending services: management
   ```

Monitoring on OpenShift

Monitor your API Connect deployment on OpenShift.

See:

- **Monitoring the API Connect cluster on OpenShift**
  Monitor the status of the API Connect cluster on OpenShift using either the command line or the OpenShift web console.
- **Monitoring Postgres disk usage on OpenShift**
  In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.
- **Monitoring platform health on Red Hat OpenShift using Kubernetes events**
  Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

Monitoring the API Connect cluster on OpenShift

Monitor the status of the API Connect cluster on OpenShift using either the command line or the OpenShift web console.

You deploy the API Connect subsystems on OpenShift with the **`APIConnect`** operator and the **`apiconnectcluster`** CR. The CR contains a status section that is regularly updated by the operator to display the current status of the all the API Connect subsystems.

When you finish deploying API Connect, you can monitor the status of **`apiconnectcluster`** using the command line or the OpenShift web console.

- Monitoring the API Connect cluster using the command line
- Monitoring the API Connect cluster using the OpenShift web console

Note: You can also monitor the disk usage of Postgres microservices. The **`APIConnect`** operator continuously tracks the disk usage of postgres microservices and provides useful information in the **`ManagementCluster`** custom resource. For information on viewing the postgres disk status, see Monitoring Postgres disk usage on OpenShift.

Monitoring the API Connect cluster using the command line

Run the following command to display the status of the API Connect cluster:

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

The response looks like the following example:

```
NAME          READY   STATUS    VERSION         RECONCILED VERSION   AGE
production    7/7     Ready     10.0.6.0    10.0.6.0-3323    24h
```

The following list explains the values in the response:

- **`NAME`**: Name of the API Connect instance (in the example, the instance name is "production").
- **`READY`**: Current number of subsystems and components that are running **/** Expected number of subsystems and components that should be running.
- **`STATUS`**: Current status of API Connect.
  If the status says **`Warning`**, then run **`oc describe apiconnectcluster -n <namespace>`** and check for more details in the **`Status`** sections. For example, a known issue on v10.0.6 and previous with WAL archiving:

  ```
  Status:
    ...
    Conditions:
      ...
      Message:                              WAL Archiving is not working. Last successful archive time: 2022-07-26
  20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
  ```

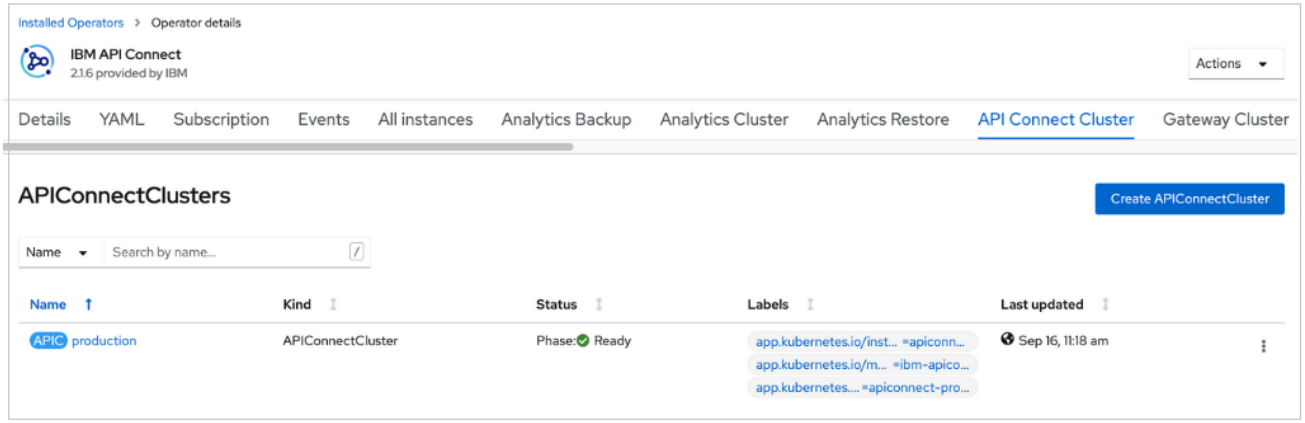
```
    Reason:                              WALArchiveNotWorking
    ...
```

Note: If you see the **`WAL Archiving is not working`** warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.

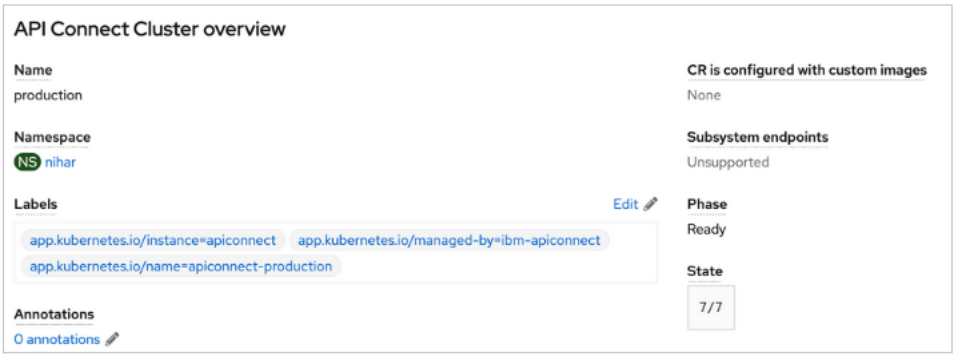
- **`VERSION`**: API Connect version specified in the CR's **`spec.version`** setting.
- **`RECONCILED VERSION`**: Current API Connect version.
- **`AGE`**: Age of the **`apiconnectcluster`** CR.

Monitoring the API Connect cluster using the OpenShift web console

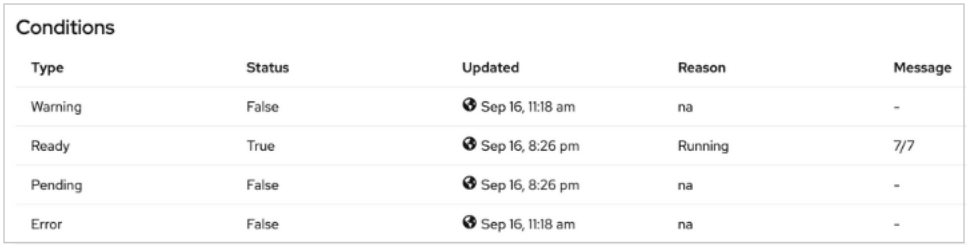
1. Start the OpenShift web console.
2. Click Installed operators.
3. Locate the namespace (project) where you installed API Connect:
   - If you installed the API Connect operator in all namespaces, select openshift-operators
   - if you installed the API Connect operator in a single namespace, select specific project
4. Hover next to the name of the API Connect namespace and click API Connect Cluster.
5. On the API Connect Clusters page, check the Status column.
   The status should display "Ready" as in the following image:



6. Click the instance name (in the example, the instance name is production to display the status overview page.
   The overview page displays high-level information about your deployment, as in the following image:



Also on the overview page, the "Conditions" section displays the list of possible service conditions, as in the following image:



A healthy deployment displays "True" for the "Ready" status.

During a fresh install, an upgrade or a day-2 operation (such as a Management backup S3 configuration change), the **`apiconnectcluster`** CR status might change to a different state depending on the action, but ultimately it should always reach the **`Ready`** status.

Monitoring Postgres disk usage on OpenShift

In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

The Postgres database is the core database used in the Management subsystem. It is important to monitor the Postgres disk usage to avoid running out of space and causing an outage in your deployment.

The **APIConnect** operator tracks the current disk usage of the Postgres components, and regularly updates the **ManagementCluster** CR's status. When one or more of the Postgres components occupy 60% of the PVC (persistent volume claim) capacity, the **APIConnect** operator changes the CR's status from **Running** to **Warning**.

Important: If the Postgres disk usage reaches 80%, then the **APIConnect** operator brings down Postgres. If this situation occurs, contact IBM Support.
Note that there are various types of storage classes which can be used to deploy Management subsystem; for example, **local-storage** or **ceph block**. When **local-storage** is used, the entire disk is allocated to the worker node. In some cases, before the **APIConnect** operator reacts to an 80% usage condition, Kubernetes itself might face disk pressure and start evicting pods. In this situation, you might want to increase the size of the disk allocated to the worker node as explained in Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database V10.0.6.0.

Viewing the current disk usage

The **ManagementCluster** instance includes the **.status.postgresDataStats** field, where the operator displays the current disk usage of Postgres components. Run the following command to get the disk usage:

```
oc get mgmt -o json | jq .items[].status.postgresDataStats
```

The response looks like the following example:

```
[
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
    "pvcCapacity": 51200,
    "pvcName": "",
    "pvcType": "WAL",
    "pvcUsed": 400,
    "pvcUsedPercentage": 8
  },
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
    "pvcCapacity": 184320,
    "pvcName": "",
    "pvcType": "PostgreSQL",
    "pvcUsed": 95,
    "pvcUsedPercentage": 8
  }
]
```

Warning condition is populated at 60% usage

When one of the Postgres components uses 60% of its allocated space, the operator changes the overall status of the **ManagementCluster** to **Warning** with an appropriate warning message. The operator also updates the **postgresDataStats** section with the current data usage of the Postgres components.

Attention: If you encounter the **Warning** condition, contact IBM Support for help correcting the root cause.
To view the status, run the following command:

```
oc get mgmt
```

The following example response shows the warning state:

```
NAME               READY   STATUS    VERSION    RECONCILED VERSION   MESSAGE
AGE
stv3-management    6/8     Warning   10.0.7.0   10.0.7.0-5363        Some services are not ready - see status condition for
details   3d20h
```

To view just the status, run the following command:

```
oc get mgmt -o json | jq .items[].status.phase
```

In this example, the status displayed in the response is "Warning":

```
"Warning"
```

To see the list of current status conditions, run the following command:

```
oc get mgmt -o json | jq .items[].status.conditions
```

The following example response displays the list of status conditions. Notice that the **Warning**"condition is set to "True" and the other conditions are set to "False":

```
status:
  conditions:
  - lastTransitionTime: "2022-06-02T19:38:09Z"
    message: Warning threshold=60%, Current disk usage=63%, Is wal archiving working?=false.
      Database shutdown starts at 80%. Please contact IBM Support immediately.
    reason: wal_disk_usage_more_than_warning_threshold
    status: "True"
    type: Warning
```

Error condition is populated at 80% usage

When one of the Postgres components uses 80% of its allocated space, the operator changes the overall status of the **ManagementCluster** to **Error** and brings down Postgres avoid problems that can occur if the disk becomes completely filled.

Attention: If you encounter the **Error** condition, contact IBM Support for help correcting the root cause.
In this case, when you run the following command to view the status conditions, you will see that the "Error" condition is set to "True:

```
oc get mgmt -o json | jq .items[].status.conditions
```

For example, the following condition shows the **Error** status set to "True" due to disk usage:

```
status:
  conditions:
- lastTransitionTime: "2022-06-03T00:40:54Z"
    message: Error threshold=80%, Current disk usage=82%, Is wal archiving working?=false.
      Database is in shutdown mode and management services are disabled. Please contact
      IBM Support immediately.
    reason: wal_disk_usage_more_than_error_threshold
    status: "True"
    type: Error
```

Monitoring platform health on Red Hat OpenShift using Kubernetes events

Monitor the health of your API Connect deployment by viewing the Kubernetes events that are generated for every status change in each subsystem CR.

What are the Kubernetes events?

The CR for every subsystem of API Connect has a status condition section, which is regularly updated by the API Connect operator with the current status. The API Connect operator additionally generates a Kubernetes event whenever there is a change in the CR status.

Events are enabled in API Connect operator for the following resources.

- **APIConnectCluster** CR
- **ManagementCluster** CR
- **GatewayCluster** CR
- **AnalyticsCluster** CR
- **PortalCluster** CR

Two types of events are generated when the CR status changes:

- **Normal** - Indicates that the status change is not causing any issues.
- **Warning** - Indicates that there a problem that requires your attention.

You can automatically monitor the status condition in your existing alerting system, or by integrating the Kubernetes events into a third-party alerting solution. For more information, see the following blog post: How to use Kubernetes events for effective alerting and monitoring.

Viewing the events

The default retention period of events in Red Hat OpenShift is 3 hours (controlled by **kube-apiserver**). To view the events, complete the following steps:

1. In the Web UI, click Operators > Installed Operators > IBM API Connect > API Connect Clusters.
2. Select an instance by clicking its name: Management cluster, Gateway cluster, Portal cluster, or Analytics cluster. If you installed using the top-level **APIConnectCluster** CR, you can additionally select API Connect cluster.
3. Select the Events tab to see all of the events that were generated for that instance.

Monitoring on Cloud Pak for Integration

Monitor your API Connect deployment on Cloud Pak for Integration.

See:

- **Monitoring the API Connect cluster on Cloud Pak for Integration**
  Monitor the status of the API Connect cluster on Cloud Pak for Integration using either the command line or the IBM Cloud Pak Platform UI.
- **Monitoring Postgres disk usage on Cloud Pak for Integration**
  In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

Monitoring the API Connect cluster on Cloud Pak for Integration

Monitor the status of the API Connect cluster on Cloud Pak for Integration using either the command line or the IBM Cloud Pak Platform UI.

In Cloud Pak for Integration, the API Management capability is provide by API Connect. You deploy the API Connect subsystems on Cloud Pak for Integration with the **APIConnect** operator and the **apiconnectcluster** CR. The CR contains a status section that is regularly updated by the operator to display the current status of the all the API Connect subsystems.

When you finish deploying API Connect, you can monitor the status of **apiconnectcluster** using the command line or the OpenShift web console.

- Monitoring the API Connect cluster using the command line
- Monitoring the API Connect cluster using the Platform UI

Note: You can also monitor the disk usage of Postgres microservices. The **APIConnect** operator continuously tracks the disk usage of postgres microservices and provides useful information in the **ManagementCluster** custom resource. For information on viewing the postgres disk status, see Monitoring Postgres disk usage on Cloud Pak for Integration.

Monitoring the API Connect cluster using the command line

Run the following command to display the status of the API Connect cluster:

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

The response looks like the following example:

```
NAME          READY   STATUS   VERSION         RECONCILED VERSION   AGE
production    7/7     Ready    10.0.6.0        10.0.6.0-3323        24h
```

The following list explains the values in the response:

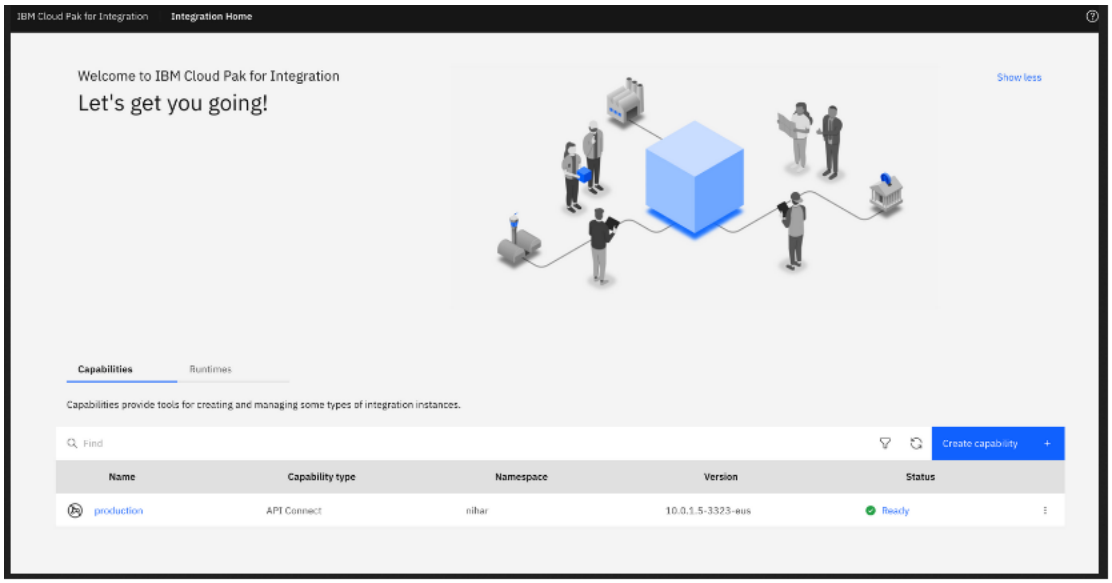
- **NAME**: Name of the API Connect instance (in the example, the instance name is "production").
- **READY**: Current number of subsystems and components that are running **/** Expected number of subsystems and components that should be running.
- **STATUS**: Current status of API Connect.
  If the status says **Warning**, then run **oc describe apiconnectcluster -n <namespace>** and check for more details in the **Status** sections. For example, a known issue on v10.0.6 and previous with WAL archiving:

  ```
  Status:
    ...
    Conditions:
      ...
      Message:                          WAL Archiving is not working. Last successful archive time: 2022-07-26
  20:37:14.456465 +0000 UTC. Last archive failed time: 2022-07-26 21:10:34.172539 +0000 UTC
      Reason:                           WALArchiveNotWorking
      ...
  ```

  Note: If you see the **WAL Archiving is not working** warning message, then verify your S3 backup settings (bucket name, folder name, credentials, certificates), gather logs and open a support request.
- **VERSION**: API Connect version specified in the CR's **spec.version** setting.
- **RECONCILED VERSION**: Current API Connect version.
- **AGE**: Age of the **apiconnectcluster** CR.

Monitoring the API Connect cluster using the Platform UI

1. Start the Platform UI.
2. Click the Integration instances tab.
3. Locate the "API Connect" capability that uses the instance name that you specified as the Name during installation.
   A healthy API Connect deployment displays "Ready" in the Status column



4. Click the status value to display information about the current conditions, as in the following image:

A healthy deployment displays "True" for the "Ready" status.

During a fresh install, an upgrade or a day-2 operation (such as a Management backup S3 configuration change), the **apiconnectcluster** CR status might change to a different state depending on the action, but ultimately it should always reach the **Ready** status.

Monitoring Postgres disk usage on Cloud Pak for Integration

In API Connect, you can monitor the disk space that is used by the Postgres database in the Management subsystem.

The Postgres database is the core database used in the Management subsystem. It is important to monitor the Postgres disk usage to avoid running out of space and causing an outage in your deployment.

The **APIConnect** operator tracks the current disk usage of the Postgres components, and regularly updates the **ManagementCluster** CR's status. When one or more of the Postgres components occupy 60% of the PVC (persistent volume claim) capacity, the **APIConnect** operator changes the CR's status from **Running** to **Warning**.

Important: If the Postgres disk usage reaches 80%, then the **APIConnect** operator brings down Postgres. If this situation occurs, contact IBM Support.
Note that there are various types of storage classes which can be used to deploy Management subsystem; for example, **local-storage** or **ceph block**. When **local-storage** is used, the entire disk is allocated to the worker node. In some cases, before the **APIConnect** operator reacts to an 80% usage condition, Kubernetes itself might face disk pressure and start evicting pods. In this situation, you might want to increase the size of the disk allocated to the worker node as explained in Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database V10.0.6.0.

Viewing the current disk usage

The **ManagementCluster** instance includes the **.status.postgresDataStats** field, where the operator displays the current disk usage of Postgres components. Run the following command to get the disk usage:

```
oc get mgmt <mgmt-cr-name> -o json | jq .status.postgresDataStats
```

The response looks like the following example:

```
[
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
    "pvcCapacity": 51200,
    "pvcName": "",
    "pvcType": "WAL",
    "pvcUsed": 400,
    "pvcUsedPercentage": 8
  },
  {
    "instanceName": "<mgmt cr name>-site1-db-1",
    "podName": "",
    "pvcCapacity": 184320,
    "pvcName": "",
    "pvcType": "PostgreSQL",
    "pvcUsed": 95,
    "pvcUsedPercentage": 8
  }
]
```

Warning condition is populated at 60% usage

When one of the Postgres components uses 60% of its allocated space, the operator changes the overall status of the **ManagementCluster** CR and the **APIConnectCluster** CR to **Warning** with an appropriate warning message. The operator also updates the **postgresDataStats** section with the current data usage of the Postgres components.

Attention: If you encounter the **Warning** condition, contact IBM Support for help correcting the root cause.
To view the status from the command line, run the following command:

- **APIConnectCluster** CR:

  ```
  oc get apiconnect
  ```

  The following example response shows the warning status:

  ```
  NAME          READY
  STATUS   VERSION        RECONCILED VERSION   AGE
  production   Current WAL disk usage of production-mgmt-31be1757-postgres is 60 percent. DATABASE SHUTDOWN starts at 80
  percent utilization. Please contact IBM Support immediately.   Warning    10.0.1.5-eus   10.0.1.5-3394-eus    11d
  ```

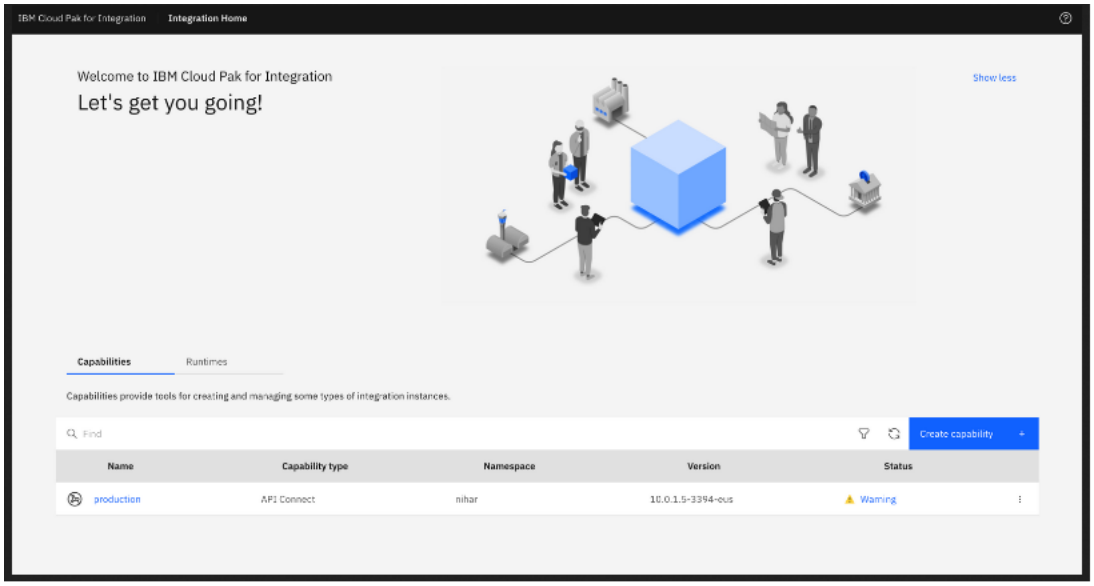
- **ManagementCluster** CR:

  ```
  oc get mgmt
  ```

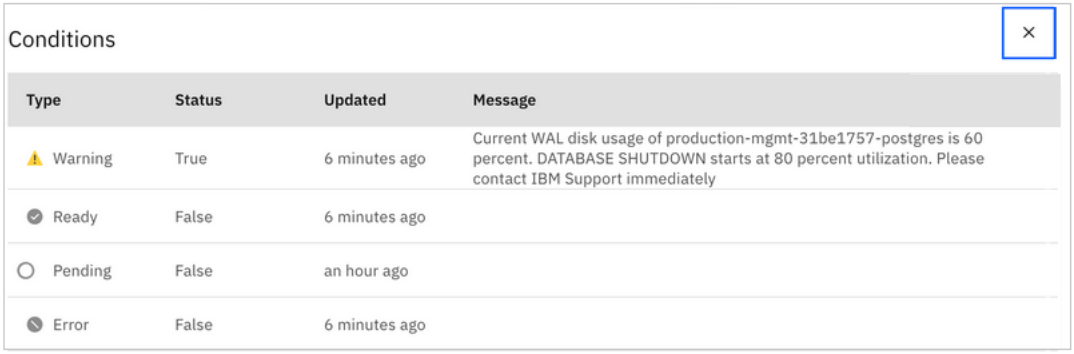
  The following example response shows the warning status:

  ```
  NAME              READY   STATUS    VERSION         RECONCILED VERSION   AGE
  production-mgmt   18/18   Warning   10.0.7.0-eus    10.0.7.0-3394-eus    43h

  Current WAL disk usage of m1-ed60c42d-postgres is 61 percent. DATABASE SHUTDOWN starts at 80 percent utilization. Please
  contact IBM Support immediately
  ```

To view the status in IBM Cloud Pak Platform UI, click the Integration instances tab to see the status displayed next to the instance name



Click the status value (Warning, for this example) to display the Conditions list where a detailed message explains why the condition was set:



Error condition is populated at 80% usage

When one of the Postgres components uses 80% of its allocated space, the operator changes the overall status of the **`ManagementCluster`** to **`Error`** and brings down Postgres avoid problems that can occur if the disk becomes completely filled.

Attention: If you encounter the **`Error`** condition, contact IBM Support for help correcting the root cause.
To view the status from the command line, run the following command:

- **`APIConnectCluster`** CR:

  ```
  oc get apiconnect
  ```

  The following example response shows the error status:

  ```
  NAME          READY
  STATUS    VERSION         RECONCILED VERSION    AGE
  production    Current disk usage is more than 80 percent. DATABASE will be shutdown. Please contact IBM Support
  immediately.    Error     10.0.1.5-eus    10.0.1.5-3394-eus     11d
  ```

- **`ManagementCluster`** CR:

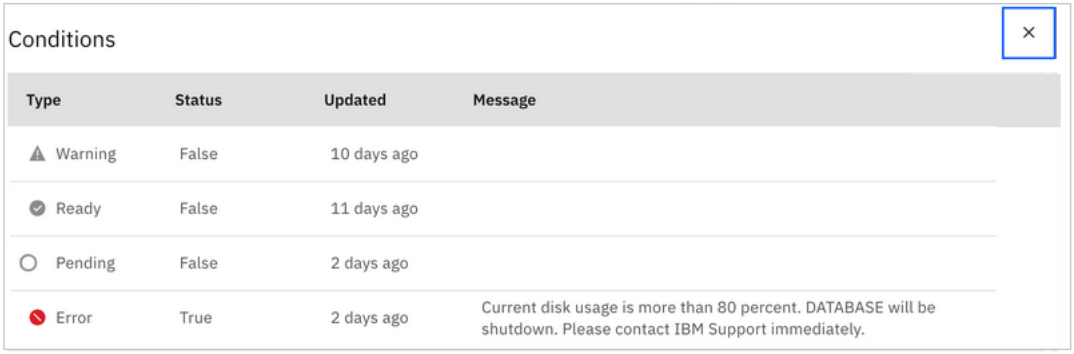
  ```
  oc get mgmt
  ```

  The following example response shows the warning status:

  ```
  NAME                READY    STATUS    VERSION         RECONCILED VERSION    AGE
  production-mgmt     9/9      Error     10.0.7.0    10.0.7.0-3394     11d
  ```

To view the status in the Platform UI, click the Integration instances tab to see the status displayed next to the instance name

Click the status value (Error in this example) to display the Conditions list where a detailed message explains why the condition was set:



Obtaining simple health check data of Developer Portal sites by using a REST API call

Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

About this task

You can use the site health REST API to determine whether a particular Developer Portal site is running. The site health API returns the current system time of the site if both the database and web server are running. This API is fast and puts no load on the system, so it is ideal for use with load balancers to help them determine where to route traffic.

Procedure

To call the site health REST API, append `/health` to the end of your Developer Portal site URL in your web browser, as follows:

`site_url/health`

Where `site_url` is the URL of the Developer Portal site that you want to check.
If both the database and web server of the site are running, the web browser returns the current system time. For example:

```
1511367695
```

If either the database or the web server of the site is not running, the web browser returns an error that the site can't be reached.

Logging

Configure logging for your deployment on Kubernetes.

See:

- **Gathering post-mortem logs**
  The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.

- **Changing logging levels**
  You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

Gathering post-mortem logs

The **generate_postmortem.sh** script gathers all logs for troubleshooting and diagnostics.

About this task

When you contact IBM Support, API Connect pod logs and associated Kubernetes or OpenShift environment data are normally required to help in diagnostics. The **generate_postmortem.sh** script gathers all the required logs and data.

The logs that are gathered by the **generate_postmortem.sh** script include the logs of the current running API Connect containers. The container logs have a maximum size, and older log messages are not retained when the maximum size is reached. To ensure that the logs gathered cover the time when the problem occurred, it is recommended to reproduce the problem and then immediately gather the logs. Also, to ensure that you have logs of previous failed containers, the recommended best practice is to offboard your logs to a remote server for long-term storage. Use ELK, or another logging infrastructure, to gather logs that include terminated pods and processes. For more information, see https://www.elastic.co/elk-stack

Procedure

1. Download the v10 **generate_postmortem.sh** script from https://github.com/ibm-apiconnect/v10-postmortem.
2. Run the script in your Kubernetes or OpenShift environment, following the instructions on https://github.com/ibm-apiconnect/v10-postmortem.
3. Upload the generated output file with your IBM support case.

What to do next

If the logs do not cover far enough back in time, or conversely if they are too large, adjust the log retention size. For further information, see: https://docs.docker.com/config/containers/logging/json-file/.

If you are not able to run the **generate_postmortem.sh** script, the API Connect container logs can be gathered manually with the following command:

```
kubectl logs -n <namespace> <pod name> [-c <container name>]
```

Redirect the output of the **logs** command to a file that you can upload to the support case, for example:

```
kubectl logs -n apic-ns apic-apim-77fdf47c55-vgnr2 > apimpod.log
```

Changing logging levels

You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

About this task

For apim-v2 pods, the default logging settings do not include entry and exit trace and logging of large payloads.

Logging of large payloads is typically needed if you are logging **apim:webhookPayload** or other processes that log large variables.

You can also set the debug level without needing to restart the pods.

You can enable the settings through either the APIM deployment YAML file, the toolkit, or the REST APIs. Note, you must use the toolkit or REST API to set the debug level without restarting the pods.

Procedure

- Updating the settings in the APIM deployment YAML file
  Note: This action causes a pod restart.
  - The DEBUG variable works the same as prior releases.

    ```
    - env:
        - name: DEBUG
          value:
    audit,bhendi:error,bhendi:probe,bhendi:flags,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:
    *
    ```

    To enable entry and exit trace, add **trace:*** to the start of the debug string:

    ```
    - env:
        - name: DEBUG
          value: trace:*,audit,bhendi:error,bhendi:probe,bhendi:flags,bhendi:webhookAudit,bhendi:cassandra-
    transactions,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*,apim:taskmanager:info
    ```

  - To enable large object logging:

    ```
    - env:
        - name: VELOX_LOG_FULL_PAYLOAD
          value: "true"
    ```

- Updating the settings by using the toolkit
  Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

  ```
  apic log-spec:get
  apic log-spec:update  LOG_SPEC_FILE
  ```

  Where the spec file contains:

  ```
  {"specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error", "large_objects": true}
  ```

- Updating the settings by using the REST API
  Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

  ```
  GET /cloud/api/log-spec
  PUT /cloud/api/log-spec
  ```

  For example, to set debug level and large objects logging:

  ```
  curl -k https://172.16.140.212:3003/api/cloud/log-spec -X PUT -H "Authorization: Bearer $BEARER"
      -H "Accept: application/json" -d '{"specification":
  "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error","large_objects": true}'
      -H "Content-Type: application/json"
  ```

  The command responds with:

  ```
  {
      "specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error",
      "large_objects": true,
      "message": "Successfully changed the log specification on all apim-v2 pods. Success on IP(s): 172.16.140.212,
  172.16.140.213"
  }
  ```

  Note: The response might include the following warning, which can be ignored:

  ```
  WARNING: Could not change the log specification on all apim-v2 pods. Success on IP(s): 4.5.6.7  Failed on IP(s): 1.2.3.4
  ```

  To view the REST APIs for logging, go to API Connect REST APIs, and select IBM API Connect Platform - Cloud Management API 2.0.0 reference > Resource: Log Spec.

Troubleshooting

Troubleshooting problems with your API Connect subsystems.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
For common operational problems with API Connect, such as UI login failure, or published APIs not running on your gateways, see: Troubleshooting API Connect.

For problems with the management subsystem database and PVC resizing, see:

- Troubleshooting the management subsystem on Kubernetes V10.0.6.0
- Troubleshooting the management subsystem on OpenShift and Cloud Pak for Integration V10.0.6.0

For problems with the analytics subsystem, see Troubleshooting analytics.

For problems with the portal subsystem, see Troubleshooting the developer portal.

To run a power-cycling check on OpenShift and Cloud Pak for Integration, see Power-cycling check on OpenShift and Cloud Pak for Integration.

- **Troubleshooting the management subsystem on Kubernetes V10.0.6.0**
  Troubleshoot problems with the management subsystem and the management subsystem database

Troubleshooting the management subsystem on Kubernetes V10.0.6.0

Troubleshoot problems with the management subsystem and the management subsystem database

These topics apply to V10.0.6.0 only:

- **Troubleshooting the management database V10.0.6.0**
  You can troubleshoot the API Connect management database
- **Increasing the PostgreSQL archive_timeout on Kubernetes V10.0.6.0**
  Increase the `archive_timeout` to `3600` (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.
- **Recovering when disks are filled by the management database V10.0.6.0**
  Resize a PersistentVolumeClaim (PVC) to recover when disks are filled by the management database.

Troubleshooting the management database V10.0.6.0

You can troubleshoot the API Connect management database

Note: The information in this topic applies to V10.0.6.0 only.

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the **patronictl reinit** command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. Exec onto the failing pod:

   ```
   kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
   ```

2. List the cluster members:

   ```
   patronictl list
   + Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
   |                          Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
   +----------------------------------------------------------+----------------+--------+--------------+----+-----------+
   |    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | start failed |    |   unknown |
   | fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
   |  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
   +----------------------------------------------------------+----------------+--------+--------------+----+-----------+
   ```

   In the example shown above **fxpk-management-01191b80-postgres-586f899fdf-6s25b** is not in running state.

   Note the **clusterName** and **replicaName** which are not up:

   - **clusterName** - **fxpk-management-01191b80-postgres**
   - **replicaName** - **fxpk-management-01191b80-postgres-586f899fdf-6s25b**

3. Run:

   ```
   patronictl reinit <clusterName> <replicaName-which-is-not-running>
   ```

   Example:

   ```
   patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
   + Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
   |                          Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
   +----------------------------------------------------------+----------------+--------+--------------+----+-----------+
   |    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | start failed |    |   unknown |
   | fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
   |  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
   +----------------------------------------------------------+----------------+--------+--------------+----+-----------+
   Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
   Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
   ```

4. Run **patronictl list** again.
   You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

   For example:

   ```
   bash-4.2$ patronictl list
   + Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+---------+----+-----------+
   |                          Member                          |      Host      |  Role  |  State  | TL | Lag in MB |
   +----------------------------------------------------------+----------------+--------+---------+----+-----------+
   |    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | running |  1 |     23360 |
   | fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader | running |  3 |           |
   |  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        | running |  3 |         0 |
   +----------------------------------------------------------+----------------+--------+---------+----+-----------+
   ```

5. The pod that previously was in an **Unknown** or **Pending state** or **(0/1) Running** state is now in **(1/1) Running** state.

Increasing the PostgreSQL `archive_timeout` on Kubernetes V10.0.6.0

Increase the **archive_timeout** to **3600** (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

Note: The information in this topic applies to V10.0.6.0 only.
PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured **archive_timeout** period elapses. By default, the **archive_timeout** is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the **archive_timeout** value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see Monitoring Postgres disk usage.

Complete the following steps to increase the PostgreSQL **archive_timeout** value.

Procedure

1. Get the name of the **pgha** config map by running the following command:

   ```
   kubectl -n <namespace> get cm -l pgha-config='true'
   ```

   The response looks like the following example:

   ```
   NAME                                              DATA   AGE
   apis-produ-d18f3f0-0d18f3f0-postgres-pgha-config   5      22d
   ```

2. Edit the config map by running the following command:

   ```
   kubectl -n <namespace> edit configmap <config-map-name>
   ```

   In the command, replace ***<config-map-name>*** with the **NAME** returned in step 1.

3. Change the **archive_timeout** value to **3600**.
   The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.

4. Save the file and exit the editor.
   The configuration is updated automatically; archives will be generated and pushed every **3600** seconds.

Recovering when disks are filled by the management database V10.0.6.0

Resize a PersistentVolumeClaim (PVC) to recover when disks are filled by the management database.

Note: The information in this topic applies to V10.0.6.0 only.
There are two methods to resize a PVC:

- Expand the PVC.
- Create a new, larger PVC, and copy the PVC contents.

Expansion of the PVC is easier and quicker than creating a new PVC and copying contents. However, some storage classes do not support PVC volume expansion. For these classes, you must create a new PVC.

Note: If you are using local storage, the only way to support increased data size is to add a disk

1. Determine if the StorageClass which provisioned the PVC supports VolumeExpansion.
   You can only expand a PVC if its storage class **allowVolumeExpansion** field is set to **true**.

   ```
   $ kubectl get sc

   NAME              PROVISIONER
   rook-ceph-block   rook-ceph.rbd.csi.ceph.com

   $ kubectl describe sc/rook-ceph-block | grep "AllowVolumeExpansion"

   AllowVolumeExpansion:  True
   ```

2. Choose a resizing method for the storage class:
   - If **AllowVolumeExpansion:** is **true**, go to Resizing a PersistentVolumeClaim for Postgres by volume expansion V10.0.6.0
   - If **AllowVolumeExpansion:** is not **true**, you cannot expand the PVC. Go to Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC V10.0.6.0.

- **Resizing a PersistentVolumeClaim for Postgres by volume expansion V10.0.6.0**
  Resize a PersistentVolumeClaim (PVC) for Postgres that supports volume expansion, on native Kubernetes.
- **Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC V10.0.6.0**
  Resize a PersistentVolumeClaim (PVC) for Postgres by creating a new, larger PVC, and moving content between the PVCs, on native Kubernetes.

Resizing a PersistentVolumeClaim for Postgres by volume expansion V10.0.6.0

Resize a PersistentVolumeClaim (PVC) for Postgres that supports volume expansion, on native Kubernetes.

Before you begin

Determine whether your storage class supports volume expansion. See Recovering when disks are filled by the management database V10.0.6.0.

About this task

Note: The information in this topic applies to V10.0.6.0 only.
You can expand a PVC if your storage provisioner supports volume expansion. This procedure is for use on native Kubernetes.

Note: This procedure is for use only with Postgres.

Procedure

1. Determine the primary Postgres pod.

   ```
   kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
   ```

2. Gracefully shutdown Postgres pods.
   a. Exec into the Postgres primary pod.

      ```
      kubectl exec -it <postgres-pod> -- bash
      ```

   b. Pause Patroni process. Note: you can skip this step if using the one replica profile.

      ```
      patronictl pause
      ```

   c. Stop the Postgres process gracefully.

      ```
      pg_ctl stop -D /pgdata/<cluster-name>
      ```

      For example:

      - `cluster-name` is usually the name of the cluster. If primary pod name is `m1-1b12bf81-postgres-58dfcc5f97-fsvvt`, `cluster-name` is `m1-1b12bf81-postgres`.
      - There will be a folder named `cluster-name` located under `/pgdata/`.
      - In this example, the command is:

        ```
        pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
        ```

3. Determine the PVC linked to the primary Postgres pod, and obtain the `claimName` of the PVC.

   ```
   kubectl get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
   ```

4. Scale down the Postgres deployment.

   ```
   kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
   ```

   If you are using `n3` profile then you find 3 deployments. Scale them down:

   ```
   kubectl scale deploy <deployment-name> --replicas=0
   ```

5. Edit the PVC you want to increase.

   ```
   kubectl edit pvc <claimName>
   ```

6. Update `spec.resources.requests.storage` field value with new size. For example:

   ```
   spec:
     resources:
       requests:
         storage: 20Gi
   ```

7. Scale up the Postgres deployment.

   ```
   kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
   ```

   If you are using the `n3` profile then you find 3 deployments. Scale them up:

   ```
   kubectl scale deploy <deployment-name> --replicas=1
   ```

8. If using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.
   a. Determine the new primary Postgres pod.

      ```
      kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
      ```

   b. Exec into the new Postgres primary pod.

      ```
      kubectl exec -it <postgres-pod> -- bash
      ```

   c. Resume Patroni.

      ```
      patronictl resume
      ```

Resizing a PersistentVolumeClaim for Postgres by moving content to a new PVC V10.0.6.0

Resize a PersistentVolumeClaim (PVC) for Postgres by creating a new, larger PVC, and moving content between the PVCs, on native Kubernetes.

Before you begin

Note: The information in this topic applies to V10.0.6.0 only.
Determine whether your storage class supports volume expansion. See Recovering when disks are filled by the management database V10.0.6.0.

About this task

Use this procedure when the storage class does not support volume expansion. This procedure is for use on native Kubernetes.

Note: This procedure is for use only with Postgres.

Procedure

1. Create a new temporary PVC. Ensure the requested storage size is more than the original PVC. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: temp-pvc
spec:
  storageClassName: rook-ceph-block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

2. Determine the primary Postgres pod.

```
kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
```

3. Gracefully shutdown Postgres pods.
   a. Exec into the Postgres primary pod.

      ```
      kubectl exec -it <postgres-pod> -- bash
      ```

   b. Pause the Patroni process. Note: you can skip this step if using the one replica profile.

      ```
      patronictl pause
      ```

   c. Stop the Postgres process gracefully.

      ```
      pg_ctl stop -D /pgdata/<cluster-name>
      ```

      For example:

      - ***cluster-name*** is usually the name of the cluster. If primary pod name is **m1-1b12bf81-postgres-58dfcc5f97-fsvvt**, ***cluster-name*** is **m1-1b12bf81-postgres**.
      - There will be a folder named ***cluster-name*** located under **/pgdata/**.
      - In this example, the command is:

        ```
        pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
        ```

4. Determine the PVC linked to the primary Postgres pod, and obtain the **claimName** of the PVC.

```
kubectl get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
```

5. Scale down the Postgres deployment.

```
kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

   If you are using **n3** profile then you find 3 deployments. Scale them down:

```
kubectl scale deploy <deployment-name> --replicas=0
```

6. Start a BusyBox pod with the source and destination PVC volumes mounted, to swap the content between PVCs. For example:

```
apiVersion: v1
kind: Pod
metadata:
  name: swap-pvc
spec:
  volumes:
    - name: volume-source
      persistentVolumeClaim:
        claimName: m1-mgmt-35817114-postgres-wal
    - name: volume-destination
      persistentVolumeClaim:
        claimName: temp-pvc
  containers:
    - name: debug
      image: busybox
      command: ['sleep', '3600']
      volumeMounts:
        - name: volume-source
          mountPath: /data-source
        - name: volume-destination
          mountPath: /data-destination
```

7. Exec to the BusyBox pod, and

```
kubectl exec -it <busybox-pod-name> -- sh
```

```
cp -r data-source/ data-destination/
```

8. Copy content to the temporary PVC. For example, using the **mountPath** values shown in Step 6:

```
cp -r data-source/ data-destination/
```

9. Delete the BusyBox pod.

```
kubectl delete pod <busybox-pod-name>
```

10. Delete the original PVC.

```
kubectl delete pvc <claimName>
```

11. Create an original PVC with the same name but with increased size. For example:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: m1-mgmt-35817114-postgres-wal
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: rook-ceph-block
  volumeMode: Filesystem
```

12. Start the BusyBox pod again and this time copy the content from destination to source PVC.

    ```
    kubectl exec -it <busybox-pod> -- sh
    ```

    ```
    cp -r data-destination/ data-source/
    ```

13. Delete the BusyBox pod.

    ```
    kubectl delete pod <busybox-pod>
    ```

14. Delete the temporary PVC that you created for swapping the content.

    ```
    kubectl delete pvc temp-pvc
    ```

15. Scale up the Postgres deployment.

    ```
    kubectl get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
    ```

    If you are using the **n3** profile then you find 3 deployments. Scale them up:

    ```
    kubectl scale deploy <deployment-name> --replicas=1
    ```

16. If using the **n3** profile, resume the Patroni process. If using the **n1** profile, skip this step.
    a. Determine the new primary Postgres pod.

    ```
    kubectl get pods --selector role=master | grep -v bootstrapselector=role=master
    ```

    b. Exec into the new Postgres primary pod.

    ```
    kubectl exec -it <postgres-pod> -- bash
    ```

    c. Resume Patroni.

    ```
    patronictl resume
    ```

Troubleshooting the management subsystem on OpenShift and Cloud Pak for Integration V10.0.6.0

Troubleshoot problems with the management subsystem and the management subsystem database.

These topics apply to V10.0.6.0 only:

Troubleshooting the management database on OpenShift and Cloud Pak for Integration V10.0.6.0

You can troubleshoot the API Connect management database by complete the following tasks.

Note: The information in this topic applies to V10.0.6.0 only.

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the **patronictl reinit** command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. Exec onto the failing pod:

   ```
   oc exec -it <postgres_replica_pod_name> -n <namespace> -- bash
   ```

2. List the cluster members:

   ```
   patronictl list
   + Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
   |                         Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
   +---------------------------------------------------------+----------------+--------+--------------+----+-----------+
   |    fxpk-management-01191b80-postgres-586f899fdf-6s25b   | 172.16.172.244 |        | start failed |    |   unknown |
   ```

```
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
```

In the example shown above **fxpk-management-01191b80-postgres-586f899fdf-6s25b** is not in running state.

Note the **clusterName** and **replicaName** which are not up:

- **clusterName** - **fxpk-management-01191b80-postgres**
- **replicaName** - **fxpk-management-01191b80-postgres-586f899fdf-6s25b**

3. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
|                         Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b   | 172.16.172.244 |        | start failed |    |   unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

4. Run **patronictl list** again.
You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+---------+----+-----------+
|                         Member                          |      Host      |  Role  |  State  | TL | Lag in MB |
+---------------------------------------------------------+----------------+--------+---------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b   | 172.16.172.244 |        | running |  1 |     23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader | running |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        | running |  3 |         0 |
+---------------------------------------------------------+----------------+--------+---------+----+-----------+
```

5. The pod that previously was in an **Unknown** or **Pending state** or **(0/1) Running** state is now in **(1/1) Running** state.

Increasing the PostgreSQL archive_timeout on OpenShift and Cloud Pak for Integration V10.0.6.0

Increase change the **archive_timeout** to **3600** (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured **archive_timeout** period elapses. By default, the **archive_timeout** is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the **archive_timeout** value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see Monitoring Postgres disk usage.

Complete the following steps to increase the PostgreSQL **archive_timeout** value.

Procedure

1. Get the name of the **pgha** config map by running the following command:

```
oc -n <APIC_namespace> get cm -l pgha-config='true'
```

The response looks like the following example:

```
NAME                                            DATA   AGE
apis-produ-d18f3f0-0d18f3f0-postgres-pgha-config   5      22d
```

2. Edit the config map by running the following command:

```
oc -n <APIC_namespace> edit configmap <config-map-name>
```

In the command, replace *<config-map-name>* with the **NAME** returned in step 1.

3. Change the **archive_timeout** value to **3600**.
The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.

4. Save the file and exit the editor.
The configuration is updated automatically; archives will be generated and pushed every **3600** seconds.

Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database V10.0.6.0

Resize a PersistentVolumeClaim (PVC) on OpenShift to recover when disks are filled by the API Connect management database.

Note: The information in this topic applies to V10.0.6.0 only.
There are two methods to resize a PVC:

- Expand the PVC.
- Create a new, larger PVC, and copy the PVC contents.

Expansion of the PVC is easier and faster than creating a new PVC and copying contents. However, some storage classes do not support PVC volume expansion. For these classes, you must create a new PVC.

Note: If you are using local storage, the only way to support increased data size is to add a disk.

1. Determine if the StorageClass which provisioned the PVC supports VolumeExpansion.
   You can only expand a PVC if its storage class **allowVolumeExpansion** field is set to **true**.

   ```
   oc get sc

   NAME               PROVISIONER
   rook-ceph-block    rook-ceph.rbd.csi.ceph.com

   oc describe sc/rook-ceph-block | grep "AllowVolumeExpansion"

   AllowVolumeExpansion:  True
   ```

2. Choose a resizing method for the storage class:
   - If **AllowVolumeExpansion:** is **true**, see Resizing a PVC for Postgres on OpenShift by using volume expansion V10.0.6.0
   - If **AllowVolumeExpansion:** is not **true**, you cannot expand the PVC. See Resizing a PVC for Postgres on OpenShift by moving content V10.0.6.0.

Resizing a PVC for Postgres on OpenShift by using volume expansion V10.0.6.0

Resize a Persistent Volume Claim (PVC) for Postgres that supports volume expansion.

Before you begin

Note: The information in this topic applies to V10.0.6.0 only.
This task only applies to the PVC used by Postgres. To determine whether your storage class supports volume expansion, see Recovering on OpenShift and Cloud Pak for Integration when disks are filled by the management database V10.0.6.0.

About this task

You can expand a PVC if your storage provisioner supports volume expansion.

Procedure

1. Determine the primary Postgres pod.

   ```
   oc get pods --selector role=master | grep -v bootstrapselector=role=master
   ```

2. Gracefully shutdown Postgres pods.
   a. Exec into the Postgres primary pod.

      ```
      oc exec -it <postgres-pod> -- bash
      ```

   b. Pause Patroni process. Note: you can skip this step if using the one replica profile.

      ```
      patronictl pause
      ```

   c. Stop the Postgres process gracefully.

      ```
      pg_ctl stop -D /pgdata/<cluster-name>
      ```

      For example:

      - ***cluster-name*** is usually the name of the cluster. If primary pod name is **m1-1b12bf81-postgres-58dfcc5f97-fsvvt**, ***cluster-name*** is **m1-1b12bf81-postgres**.
      - There will be a folder named ***cluster-name*** located under **/pgdata/**.
      - In this example, the command is:

        ```
        pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
        ```

3. Determine the PVC linked to the primary Postgres pod, and obtain the **claimName** of the PVC.

   ```
   oc get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
   ```

4. Scale down the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using **n3** profile then you find 3 deployments. Scale them down:

```
oc scale deploy <deployment-name> --replicas=0
```

5. Edit the PVC you want to increase.

```
oc edit pvc <claimName>
```

6. Update **spec.resources.requests.storage** field value with new size. For example:

```
spec:
  resources:
    requests:
      storage: 20Gi
```

7. Scale up the Postgres deployment.

```
oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
```

If you are using the **n3** profile then you find 3 deployments. Scale them up:

```
oc scale deploy <deployment-name> --replicas=1
```

8. If you are using the **n3** profile, resume the Patroni process. If using the **n1** profile, skip this step.
   a. Determine the new primary Postgres pod.

   ```
   oc get pods --selector role=master | grep -v bootstrapselector=role=master
   ```

   b. Exec into the new Postgres primary pod.

   ```
   oc exec -it <postgres-pod> -- bash
   ```

   c. Resume Patroni.

   ```
   patronictl resume
   ```

Resizing a PVC for Postgres on OpenShift by moving content V10.0.6.0

Resize a Persistent Volume Claim (PVC) for Postgres on OpenShift by creating a new, larger PVC, and then moving content between the PVCs.

Before you begin

Note: The information in this topic applies to V10.0.6.0 only.
This task only applies to the PVC used by Postgres.

About this task

Use this procedure when the storage class does not support volume expansion.

Procedure

1. Create a new temporary PVC. Ensure the requested storage size is more than the original PVC. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: temp-pvc
spec:
  storageClassName: rook-ceph-block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

2. Determine the primary Postgres pod.

```
oc get pods --selector role=master | grep -v bootstrapselector=role=master
```

3. Gracefully shutdown Postgres pods.
   a. Exec into the Postgres primary pod.

   ```
   oc exec -it <postgres-pod> -- bash
   ```

   b. Pause the Patroni process. Note: you can skip this step if using the one replica profile.

   ```
   patronictl pause
   ```

   c. Stop the Postgres process gracefully.

   ```
   pg_ctl stop -D /pgdata/<cluster-name>
   ```

   For example:

- ***cluster-name*** is usually the name of the cluster. If primary pod name is **m1-1b12bf81-postgres-58dfcc5f97-fsvvt**, ***cluster-name*** is **m1-1b12bf81-postgres**.
- There will be a folder named ***cluster-name*** located under **/pgdata/**.
- In this example, the command is:

  ```
  pg_ctl stop -D /pgdata/m1-1b12bf81-postgres
  ```

4. Determine the PVC linked to the primary Postgres pod, and obtain the **claimName** of the PVC.

   ```
   oc get pod <postgres-pod> -o json | jq '.spec.volumes[] | select ( has ("persistentVolumeClaim"))'
   ```

5. Scale down the Postgres deployment.

   ```
   oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
   ```

   If you are using **n3** profile then you find 3 deployments. Scale them down:

   ```
   oc scale deploy <deployment-name> --replicas=0
   ```

6. Start a BusyBox pod with the source and destination PVC volumes mounted, to swap the content between PVCs. For example:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     name: swap-pvc
   spec:
     volumes:
       - name: volume-source
         persistentVolumeClaim:
           claimName: m1-mgmt-35817114-postgres-wal
       - name: volume-destination
         persistentVolumeClaim:
           claimName: temp-pvc
     containers:
       - name: debug
         image: busybox
         command: ['sleep', '3600']
         volumeMounts:
           - name: volume-source
             mountPath: /data-source
           - name: volume-destination
             mountPath: /data-destination
   ```

7. Exec to the BusyBox pod, and run the following commands:

   ```
   oc exec -it <busybox-pod-name> -- sh
   ```

   ```
   cp -r data-source/ data-destination/
   ```

8. Copy content to the temporary PVC. For example, using the **mountPath** values shown in Step 6:

   ```
   cp -r data-source/ data-destination/
   ```

9. Delete the BusyBox pod.

   ```
   oc delete pod <busybox-pod-name>
   ```

10. Delete the original PVC.

    ```
    oc delete pvc <claimName>
    ```

11. Create an original PVC with the same name but with increased size. For example:

    ```
    kind: PersistentVolumeClaim
    apiVersion: v1
    metadata:
      name: m1-mgmt-35817114-postgres-wal
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 20Gi
      storageClassName: rook-ceph-block
      volumeMode: Filesystem
    ```

12. Run the following commands to start the BusyBox pod again and copy the content from the destination to the source PVC:

    ```
    oc exec -it <busybox-pod> -- sh
    ```

    ```
    cp -r data-destination/ data-source/
    ```

13. Delete the BusyBox pod.

    ```
    oc delete pod <busybox-pod>
    ```

14. Delete the temporary PVC that you created for swapping the content.

    ```
    oc delete pvc temp-pvc
    ```

15. Scale up the Postgres deployment.

    ```
    oc get deploy | grep postgres | grep -v bouncer | grep -v bootstrap | grep -v backrest | grep -v operator
    ```

If you are using the `n3` profile then you find 3 deployments. Scale them up:

```
oc scale deploy <deployment-name> --replicas=1
```

16. If using the `n3` profile, resume the Patroni process. If using the `n1` profile, skip this step.
    a. Determine the new primary Postgres pod.

    ```
    oc get pods --selector role=master | grep -v bootstrapselector=role=master
    ```

    b. Exec into the new Postgres primary pod.

    ```
    oc exec -it <postgres-pod> -- bash
    ```

    c. Resume Patroni.

    ```
    patronictl resume
    ```

Power-cycling check on OpenShift and Cloud Pak for Integration

If you cycle the power on an API Connect cluster that is deployed on OpenShift or Cloud Pak for Integration, you might need to reset the IP address for the `kubernetes.io/hostname`.

About this task

You will know there is a problem if you cycle the power for the cluster and receive an error, or find that the deployment remains in the `pending` state indefinitely because some of the pods failed to start. Complete the following steps to resolve the problem:

Procedure

1. Run the following command to check the label on the worker nodes:

   ```
   oc get nodes -l kubernetes.io/hostname=localhost
   ```

   If the response shows the following message, then the power cycle did not cause a problem and you can skip the rest of this task:

   ```
   No resources found
   ```
   If the response looks like the following example (the IP address will be different), proceed to the next step:

   ```
   NAME                                         STATUS   ROLES    AGE   VERSION
   ip-233.252.0.0.us-west-1.compute.internal    Ready    worker   27h   v1.19.0+8d12420
   ```

2. Run the following command to correct the problem:

   ```
   for i in $(oc get nodes -l kubernetes.io/hostname=localhost|grep worker|awk '{print $1}'); do echo $i;  oc label nodes $i
   kubernetes.io/hostname=`echo $i|awk -F'.' '{print $1}'` --overwrite=true; done
   ```

Uninstalling API Connect

You can uninstall a Kubernetes deployment of API Connect.

About this task

You can choose to uninstall individual API Connect subsystems or you can uninstall the complete API Connect deployment. You can delete Custom Resource Definitions, Persistent Volumes, service accounts, secrets, operators and the namespace. Use one of the following topics.
Note: For information about how to maintain a two data center deployment, see Maintaining a two data center deployment.

- **Uninstalling API Connect subsystems**
  You can individually uninstall API Connect subsystems that have been deployed on Kubernetes.
- **Removing API Connect**
  You can remove all parts of the API Connect deployment on Kubernetes.
- **Removing API Connect on OpenShift or Cloud Pak for Integration**
  Remove your entire API Connect deployment on OpenShift or Cloud Pak for Integration.

Uninstalling API Connect subsystems

You can individually uninstall API Connect subsystems that have been deployed on Kubernetes.

Before you begin

If you want to disable the analytics subsystem without uninstalling it, see Shutting down the analytics subsystem.

About this task

- To delete the API Connect deployment in a Kubernetes runtime environment, you must use the Kubernetes command-line tool, kubectl.

- The API Connect operator and DataPower operator pods are needed to uninstall subsystems individually.
- If need to retain data to re-use during a re-installation later, follow the backup and restore procedures Backing up, restoring, and disaster recovery.
- If you want to delete **all** of API Connect, do not use this topic. Instead, see Removing API Connect.

Procedure

1. Uninstall the Management subsystem
   a. Save the management subsystem name:

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

      This command returns the YAML format of the management custom resource. Make a note of the name.

   b. Delete the Management Subsystem.

      ```
      kubectl delete mgmt --all -n <namespace>
      ```

      Important:
      If your deployment uses default namespaces be cautious when using **`kubectl delete <entity> --all`** commands. There can be other pods, services, and more in the default namespace. When using the default namespace, keep in mind that using **`kubernetes delete <entity> --all`**, where <*entity*> can delete non-APIC entities, such as **`sa`** (Service Accounts), **`secrets`**, **`rolebindings`**, etc.

      The Management CR deletion will take some time because the apiconnect operator must delete the Postgres cluster.

   c. Manually delete secrets and PVCs:
      i. Get all of the secrets that use the management subsystem name (which you determined in Step 1.a):

         ```
         kubectl get secrets -n <namespace> | grep <management-subsystem-name>
         ```

      ii. For each secret that uses the management subsystem name, run the following command to delete the secret:

         ```
         kubectl delete secrets <secret_name> -n <namespace>
         ```

      iii. Get all the PVCs that use the management subsystem name:

         ```
         kubectl get pvc -n <namespace> | grep <management-subsystem-name>
         ```

      iv. For each PVC that uses the management subsystem name, run the following command to delete the PVC:

         ```
         kubectl delete pvc <pvc_name> -n <namespace>
         ```

   d. If management subsystem backup and restore CRs are left stale, delete them also:
      i. Get the names of the backup CRs:

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

      ii. Delete the backup CRs:

         ```
         kubectl delete Backup <backup_cr_name> -n <namespace>
         ```

      iii. Get the names of the restore CRs:

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

      iv. Delete the restore CRs:

         ```
         kubectl delete ManagementRestore <restore_cr_name> -n <namespace>
         ```

   Note: If you have management backup CRs from V10.0.6.0 or earlier, delete those too:
      i. Get the name of the backup CR:

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

      ii. Delete the backup CR:

         ```
         kubectl delete mgmtb <backup_cr_name> -n <namespace>
         ```

      iii. Get the name of the restore CR:

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

      iv. Delete the restore CR:

         ```
         kubectl delete mgmtr <restore_cr_name> -n <namespace>
         ```

2. Uninstall Analytics subsystem
   a. Delete the Analytics subsystem, the Analytics backup, and the Analytics restore CRs.

      ```
      kubectl delete a7s --all -n <namespace>
      kubectl delete a7b --all -n <namespace>
      kubectl delete a7r --all -n <namespace>
      ```

   b. Manually delete the Analytics secrets:

      ```
      kubectl delete secrets <secret_name> -n <namespace>
      ```

      for each of the following secrets:

      ```
      analytics-ai-endpoint
      analytics-ca
      ```

```
analytics-client
analytics-server
```

c. Manually delete analytics Persistent Volume Claims (PVCs):

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
data-analytics-storage
```

3. Uninstall the Developer Portal subsystem
   a. Delete the Developer Portal subsystem, and the Portal backup and restore CRs.

```
kubectl delete ptl --all -n <namespace>
kubectl delete pb --all -n <namespace>
kubectl delete pr --all -n <namespace>
```

   b. Manually delete Developer Portal secrets:

```
kubectl delete secrets <secret_name> -n <namespace>
```

for secrets with the following naming pattern:

```
portal-admin
portal-admin-client
portal-ca
portal-client
portal-db-ca
portal-encryption-key-...
portal-server
portal-web
```

   c. Manually delete these Developer Portal PVCs:

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
admin-portal-...-www-...
backup-portal-...-www-...
db-portal-...-db-...
dblogs-portal-...-db-...
web-portal-...-www-...
```

4. Delete the DataPower Gateway subsystem.

```
kubectl delete gw --all -n <namespace>
```

Note: The following entries include both Multi-Protocol Gateway Service (v5-compatible) and API Gateway settings. Names that include "gwv5" refer to the Multi-Protocol Gateway Service (v5-compatible), and names that include "gwv6" refer to the API Gateway.

   a. Delete Service Accounts.

```
kubectl delete sa <service_account> -n <namespace>
```

for each of the following Service Accounts:

```
gwv5-datapower
gwv6-datapower
```

   b. Delete Secrets.

```
kubectl delete secrets <secret_name> -n <namespace>
```

for secrets with the following naming pattern:

```
gwv5-datapower-token-...
gwv5-dpm-sa-token-...
gwv5-endpoint
gwv5-manager-endpoint
gwv6-datapower-token-...
gwv6-dpm-sa-token-...
gwv6-endpoint.
gwv6-manager-endpoint
```

   c. Delete PVCs.

```
kubectl delete pvc <pvc_name> -n <namespace>
```

for PVCs with the following naming pattern:

```
gwv5...
gwv6...
```

- **Removing Management subsystem postgres resources on Kubernetes V10.0.6.0**
  V10.0.6.0 only: When you remove the API Connect Management subsystem by deleting its files, some postgres resources from Crunchy Data remain, and must be deleted manually before you attempt to re-install the Management subsystem.

Removing Management subsystem postgres resources on Kubernetes V10.0.6.0

V10.0.6.0 only: When you remove the API Connect Management subsystem by deleting its files, some postgres resources from Crunchy Data remain, and must be deleted manually before you attempt to re-install the Management subsystem.

About this task

Sometimes when you remove the Management subsystem, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the **kubectl get** command and specify **vendor=crunchydata**; as shown in the following example.

```
$ kubectl get pods -n apic -l vendor=crunchydata
NAME                                                     READY   STATUS      RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj      0/1     Completed   0          17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc   1/1     Running     0          2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5      0/1     Completed   0          164m

$ kubectl get deployments -n apic -l vendor=crunchydata
NAME                                                  READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo   1/1     1            1           2d21h

$ kubectl get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get jobs -n apic -l vendor=crunchydata
NAME                                                COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres   1/1           10s        17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup   1/1           37s        164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg       1/1           20s        14m

$ kubectl get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get secrets -n apic -l vendor=crunchydata
NAME                                                  TYPE     DATA   AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config        Opaque   9      6d23h
production-mgmt-7bc754c7-postgres-postgres-secret             Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret          Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-testuser-secret             Opaque   2      6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config        Opaque   9      6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret             Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret          Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret             Opaque   2      6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config   Opaque   9      2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret        Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret     Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret        Opaque   2      2d21h

$ kubectl get cm -n apic -l vendor=crunchydata
NAME                                               DATA   AGE
production-mgmt-mymgmtsite123-postgres-config          0      2d21h
production-mgmt-mymgmtsite123-postgres-leader          0      2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm    2      2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config     5      2d21h
```

Procedure

For each type of postgres resource, check to see if any resources are available and if so, delete the resources:

- pods:

  ```
  kubectl get pods -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pods -n <namespace> -l vendor=crunchydata
  ```

- deployments:

  ```
  kubectl get deployments -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete deployments -n <namespace> -l vendor=crunchydata
  ```

- service accounts:

  ```
  kubectl get sa -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete sa -n <namespace> -l vendor=crunchydata
  ```

- jobs:

  ```
  kubectl get jobs -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete jobs -n <namespace> -l vendor=crunchydata
  ```

- roles:

  ```
  kubectl get roles -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete roles -n <namespace> -l vendor=crunchydata
  ```

- rolebindings:

```
kubectl get rolebindings -n <namespace> -l vendor=crunchydata
```

```
kubectl delete rolebindings -n <namespace> -l vendor=crunchydata
```

- secrets:

  ```
  kubectl get secrets -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete secrets -n <namespace> -l vendor=crunchydata
  ```

- cm:

  ```
  kubectl get cm -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete cm -n <namespace> -l vendor=crunchydata
  ```

- pgtasks:

  ```
  kubectl get pgtasks.crunchydata.com -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pgtasks.crunchydata.com -n <namespace>
  ```

- pgreplica:

  ```
  kubectl get pgreplica -n <management_namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pgreplica -n <management_namespace> -l vendor=crunchydata
  ```

Removing API Connect

You can remove all parts of the API Connect deployment on Kubernetes.

About this task

These instructions completely remove API Connect, including PVCs and secrets.

- To delete the API Connect deployment in a Kubernetes runtime environment, you must use the Kubernetes command-line tool, kubectl.
- You must run the commands in the order shown in the procedure on this page. If you run commands out of order, you might encounter a problem where the deletion of namespaces hangs.
- If you want to retain the PVCs, which contain some of the data you built up when using the application, you must also keep the associated secret used to decrypt the data. To do this, you must backup each subsystem, and then uninstall the subsystems individually. In this case, do not use the instructions on this page. Instead, see Uninstalling API Connect subsystems.

Procedure

1. Delete all API Connect systems.

   ```
   kubectl delete apic --all -n <namespace>
   ```

   This command is the equivalent of deleting each of the individual subsystems one at a time, and also deleting all of their backup and restore CRs. The commands that run during this step are:

   - Subsystem CRs. For example:

     ```
     kubectl delete mgmt --all -n <namespace>
     kubectl delete a7s --all -n <namespace>
     kubectl delete ptl --all -n <namespace>
     kubectl delete gwv6 --all -n <namespace>
     ```

   - Backup and restore CRs:

     ```
     kubectl delete Backup --all -n <namespace>
     kubectl delete mgmtr --all -n <namespace>
     kubectl delete pb --all -n <namespace>
     kubectl delete pr --all -n <namespace>
     kubectl delete a7b --all -n <namespace>
     kubectl delete a7r --all -n <namespace>
     ```

     Note: If your deployment was previously on V10.0.6.0 or earlier, delete any pre-V10.0.7.0 management backup or restore CRs:

     ```
     kubectl delete mgmtb --all -n <namespace>
     kubectl delete mgmtr --all -n <namespace>
     ```

2. Delete the Operator deployments.

   ```
   kubectl delete deployment datapower-operator ibm-apiconnect -n <namespace>
   ```

3. Delete all Service Accounts in the namespace.
   Important:
   If your deployment uses default namespaces be cautious when using **`kubectl delete <entity> --all`** commands. There can be other pods, services, and more in the default namespace. When using the default namespace, keep in mind that using **`kubernetes delete <entity> --all`**, where <*entity*> can delete non-APIC entities, such as **`sa`** (Service Accounts), **`secrets`**, **`rolebindings`**, etc.

   ```
   kubectl delete sa --all -n <namespace>
   ```

4. Delete all Secrets in the namespace.

```
kubectl delete secrets --all -n <namespace>
```

5. Delete all Rolebindings in the namespace.

```
kubectl delete rolebindings --all -n <namespace>
```

6. Delete all Roles in the namespace.

```
kubectl delete roles --all -n <namespace>
```

7. Clean up any remaining jobs.

```
kubectl delete jobs --all -n <namespace>
```

8. Delete all PVCs in the namespace.

```
kubectl delete pvc --all -n <namespace>
```

9. Delete Issuers in the namespace.
   Note that there might be none found.

```
kubectl delete issuer --all -n <namespace>
```

10. Delete certificates in the namespace.
    Note that there might be none found.

```
kubectl delete certificates --all -n <namespace>
```

11. If you are not using the default namespace, delete the Namespace
    Important: If your deployment uses the default namespace, do not delete it. There can be other pods, services, and more in the default namespace.

```
kubectl delete ns <namespace>
```

12. Delete `validatingwebhookconfiguration`

    a. `kubectl get validatingwebhookconfiguration | awk /<ns>/'{print $1}' | xargs kubectl delete validatingwebhookconfiguration`

    b. `kubectl delete validatingwebhookconfiguration managements.validator.subsystem.apiconnect.ibm.com`

13. Delete `mutatingwebhookconfiguration`

    a. `kubectl get mutatingwebhookconfiguration | awk /$NS/'{print $1}' | xargs kubectl delete mutatingwebhookconfiguration`

    b. `kubectl get mutatingwebhookconfiguration | awk /apicop-management-defaulter/'{print $1}' | xargs kubectl delete mutatingwebhookconfiguration`

14. Delete the Apiconnect CRDs (custom resource definitions)

```
kubectl delete crd analyticsbackups.analytics.apiconnect.ibm.com
kubectl delete crd analyticsclusters.analytics.apiconnect.ibm.com
kubectl delete crd analyticsrestores.analytics.apiconnect.ibm.com
kubectl delete crd datapowerservices.datapower.ibm.com
kubectl delete crd gatewayclusters.gateway.apiconnect.ibm.com
kubectl delete crd backups.management.apiconnect.ibm.com
kubectl delete crd managementclusters.management.apiconnect.ibm.com
kubectl delete crd managementrestores.management.apiconnect.ibm.com
kubectl delete crd natsclusters.nats.io
kubectl delete crd natsserviceroles.nats.io
kubectl delete crd natsstreamingclusters.streaming.nats.io
kubectl delete crd portalbackups.portal.apiconnect.ibm.com
kubectl delete crd portalclusters.portal.apiconnect.ibm.com
kubectl delete crd portalrestores.portal.apiconnect.ibm.com
```

    Note: If your deployment was previously on V10.0.6.0 or earlier, delete any pre-V10.0.7.0 management backup or restore CRs:

```
kubectl delete crd managementbackups.management.apiconnect.ibm.com
```

15. Delete EDB CRDs:
    a. Get a list of the EDB CRDs:

```
kubectl -n <namespace> get crds | grep enterprisedb
```

       Expected output:

```
backups.postgresql.k8s.enterprisedb.io                            2023-07-26T14:09:51Z
clusters.postgresql.k8s.enterprisedb.io                           2023-07-26T14:09:51Z
poolers.postgresql.k8s.enterprisedb.io                            2023-07-26T14:09:59Z
scheduledbackups.postgresql.k8s.enterprisedb.io                   2023-07-26T14:10:02Z
```

    b. Delete the EDB CRDs:

```
kubectl -n <namespace> delete crd <crd name>
```

       where *<crd name>* is the name of each CRD output in step 15.a.

16. v10.0.6 only: Delete Crunchy CRDs

```
kubectl delete crd pgclusters.crunchydata.com
kubectl delete crd pgpolicies.crunchydata.com
kubectl delete crd pgreplicas.crunchydata.com
kubectl delete crd pgtasks.crunchydata.com
```

17. If you installed cert-manager and want to remove it now, complete the following steps:

Warning: If you are using shared clusters for API Connect with a common cert manager, be aware that this step removes the components of the cert manager from cluster wide scope.

a. Delete the cert-manager

```
kubectl delete deploy --all -n cert-manager
kubectl delete service --all -n cert-manager
kubectl delete sa -n cert-manager --all
kubectl delete secrets -n cert-manager --all
kubectl delete APIService v1beta1.webhook.cert-manager.io -n cert-manager
kubectl delete MutatingWebhookConfiguration cert-manager-webhook -n cert-manager
kubectl delete ValidatingWebhookConfiguration cert-manager-webhook -n cert-manager
kubectl delete ns cert-manager
```

b. Delete cert-manager CRDs

- Delete the cert-manager CRDs with the following:

```
kubectl delete crd issuers.cert-manager.io
kubectl delete crd certificaterequests.cert-manager.io
kubectl delete crd certificates.cert-manager.io
kubectl delete crd challenges.cert-manager.io
kubectl delete crd clusterissuers.cert-manager.io
kubectl delete crd orders.cert-manager.io
```

- Older API Connects versions may have cert-manager CRDs from an older version of cert-manager which had a different name, delete these CRDs with the following:

```
kubectl delete crd certificaterequests.certmanager.k8s.io
kubectl delete crd certificates.certmanager.k8s.io
kubectl delete crd challenges.acme.certmanager.k8s.io
kubectl delete crd clusterissuers.certmanager.k8s.io
kubectl delete crd issuers.certmanager.k8s.io
kubectl delete crd orders.acme.certmanager.k8s.io
```

- **Removing postgres resources on Kubernetes after removing API Connect V10.0.6.0**
  V10.0.6.0 only: When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

Removing postgres resources on Kubernetes after removing API Connect V10.0.6.0

V10.0.6.0 only: When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

About this task

Sometimes when you remove API Connect, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the **kubectl get** command and specify **vendor=crunchydata** as shown in the following example.

```
$ kubectl get pods -n apic -l vendor=crunchydata
NAME                                                      READY   STATUS      RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj      0/1     Completed   0          17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc   1/1     Running     0          2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5      0/1     Completed   0          164m

$ kubectl get deployments -n apic -l vendor=crunchydata
NAME                                                 READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo   1/1     1            1           2d21h

$ kubectl get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get jobs -n apic -l vendor=crunchydata
NAME                                              COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres   1/1           10s        17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup   1/1           37s        164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg       1/1           20s        14m

$ kubectl get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ kubectl get secrets -n apic -l vendor=crunchydata
NAME                                                    TYPE     DATA   AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config         Opaque   9      6d23h
production-mgmt-7bc754c7-postgres-postgres-secret              Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret           Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-testuser-secret              Opaque   2      6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config         Opaque   9      6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret              Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret           Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret              Opaque   2      6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config    Opaque   9      2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret         Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret      Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret         Opaque   2      2d21h
```

```
$ kubectl get cm -n apic -l vendor=crunchydata
NAME                                                DATA   AGE
production-mgmt-mymgmtsite123-postgres-config          0      2d21h
production-mgmt-mymgmtsite123-postgres-leader          0      2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm    2      2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config     5      2d21h
```

Procedure

For each type of postgres resources, check to see if any resources are available and if so, delete the resources:

- pods:

  ```
  kubectl get pods -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pods -n <namespace> -l vendor=crunchydata
  ```

- deployments:

  ```
  kubectl get deployments -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete deployments -n <namespace> -l vendor=crunchydata
  ```

- Jobs:

  ```
  kubectl get jobs -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete jobs -n <namespace> -l vendor=crunchydata
  ```

- roles:

  ```
  kubectl get roles -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete roles -n <namespace> -l vendor=crunchydata
  ```

- rolebindings:

  ```
  kubectl get rolebindings -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete rolebindings -n <namespace> -l vendor=crunchydata
  ```

- secrets:

  ```
  kubectl get secrets -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete secrets -n <namespace> -l vendor=crunchydata
  ```

- cm:

  ```
  kubectl get cm -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete cm -n <namespace> -l vendor=crunchydata
  ```

- pgtasks:

  ```
  kubectl get pgtasks.crunchydata.com -n <namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pgtasks.crunchydata.com -n <namespace>
  ```

- pgreplica:

  ```
  kubectl get pgreplica -n <management_namespace> -l vendor=crunchydata
  ```

  ```
  kubectl delete pgreplica -n <management_namespace> -l vendor=crunchydata
  ```

Removing API Connect on OpenShift or Cloud Pak for Integration

Remove your entire API Connect deployment on OpenShift or Cloud Pak for Integration.

About this task

These instructions completely remove API Connect, including PVCs and secrets.

Attention: Run the commands in the sequence shown on this page. If you run commands out of sequence, you might encounter a problem where the deletion of namespaces hangs.

Procedure

1. If Event Endpoint Management is installed, run the following command to determine the name of your Event Endpoint Management instance:

   ```
   oc get eem -n <APIC-namespace>
   ```

2. If you are using a top-level CR, run the following command to determine the name of your API Connect instance:

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

3. If you are using a top-level CR, run the following command to delete the API Connect operands (subsystems):

```
oc delete apiconnectcluster <name_of_apic_instance> -n <APIC-namespace>
```

4. Run the following command to delete the API Connect backup and restore CRs:

   ```
   oc delete apic --all -n <APIC-namespace>
   ```

5. Run the following command to delete the secrets that are associated with your deleted API Connect cluster:

   ```
   oc get secrets --no-headers -n <APIC-namespace> | cut -d' ' -f1 | grep "^<first_10_characters_of_apic_instance_name>-" |
   xargs oc delete secret -n <APIC-namespace>
   ```

   Where *<first_10_characters_of_apic_instance_name>* refers to the name of your API Connect instance.

   Note: This step assumes that any secrets that you want to keep in your namespace do not have a name that starts with the same 10 first characters of the API Connect Cluster name.
6. If Event Endpoint Management is installed, run the following command to delete the secrets that are associated with your deleted Event Endpoint Management instance:

   ```
   oc get secrets --no-headers -n <APIC-namespace> | cut -d' ' -f1 | grep "^<first_10_characters_of_eem_instance_name>-" |
   xargs oc delete secret -n <APIC-namespace>
   ```

   Where *<first_10_characters_of_eem_instance_name>* refers to the name of your Event Endpoint Manager instance.

   Note: This step assumes that any secrets that you want to keep in your namespace do not have a name that starts with the same 10 first characters of the Event Endpoint Manager name.
7. V10.0.6.0 only: If the only software that is deployed in your namespace is API Connect, run the following command to delete the PVCs associated with your deleted API Connect cluster:

   ```
   oc delete pvc -n <APIC-namespace> -l app.kubernetes.io/managed-by=ibm-apiconnect
   ```

8. V10.0.6.0 only: If you have no other software in your namespace that uses a CrunchyData Postgres cluster, delete the CrunchyData Postgres cluster:

   ```
   oc delete pvc -n <APIC-namespace> -l vendor=crunchydata
   ```

9. V10.0.6.0 only: Check for and remove any other Crunchy data: Removing postgres resources on OpenShift and Cloud Pak for Integration V10.0.6.0.
10. Run the following commands to delete the API Connect secret rotation CRs for the cluster:

    ```
    oc delete mgmtsr -n <APIC-namespace>  --all
    oc delete ptlsr -n <APIC-namespace> --all
    ```

11. Optional: If you created a namespace specifically for API Connect, and are not using it for anything else, you can delete the namespace. If your API Connect operators are installed in this namespace, instead of in a shared global namespace, they are also deleted.

    ```
    oc delete ns <APIC-namespace>
    ```

    Deleting the namespace removes any remaining certificates and secrets.
    Note: Namespace deletion can get stuck, check that the namespace is deleted with:

    ```
    oc get ns <APIC-namespace>
    ```

    If the namespace is still present, to see why deletion is stuck, run:

    ```
    oc describe ns <APIC-namespace>
    ```

12. Optional: Delete the API Connect operator deployments.
    If your API Connect operator deployments are in a global namespace they are not deleted when your API Connect namespace is deleted. To delete the API Connect operators:
    a. Open the OpenShift web console and go to Installed Operators, in the namespace where the API Connect operators are installed.
    b. For each operator, click the options menu, and select Uninstall Operator.
    API Connect operator might have created a number of operand requests with IBM Cloud Pak foundational services (previously called Common Services) to deploy services such as Cert-Manager. In addition, API Connect requires the use of a DataPower Gateway, so the DataPower operator might be installed in the API Connect namespace. If those artifacts are no longer needed, uninstall them individually.
    Note: DataPower can be deployed on its own without API Connect. It is possible that the DataPower operator was installed for another purpose and not for use with API Connect, so check with your organization before you delete it.
    Deleting the operators from the web console ensures that related subscriptions, CSVs, and install plans are also removed.

Removing postgres resources on OpenShift and Cloud Pak for Integration V10.0.6.0

V10.0.6.0 only: When you remove API Connect by deleting its files, some of the postgres resources from Crunchy Data remain and must be deleted manually before you attempt to re-install API Connect.

About this task

Sometimes when you remove API Connect, some postgres resources remain and must be deleted manually.

To determine whether you need to delete any postgres resources, use the **oc get** command and specify **vendor=crunchydata** as shown in the following example.

```
$ oc get pods -n apic -l vendor=crunchydata
NAME                                                        READY   STATUS      RESTARTS   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres-k77xj      0/1     Completed   0          17h
production-mgmt-mymgmtsite123-postgres-backrest-shared-repk4kzc   1/1     Running     0          2d21h
production-mgmt-mymgmtsite123-postgres-full-sch-backup-9ktp5      0/1     Completed   0          164m

$ oc get deployments -n apic -l vendor=crunchydata
NAME                                                   READY   UP-TO-DATE   AVAILABLE   AGE
production-mgmt-mymgmtsite123-postgres-backrest-shared-repo   1/1     1            1           2d21h
```

```
$ oc get sa -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get jobs -n apic -l vendor=crunchydata
NAME                                                COMPLETIONS   DURATION   AGE
backrest-backup-production-mgmt-mymgmtsite123-postgres   1/1           10s        17h
production-mgmt-mymgmtsite123-postgres-full-sch-backup   1/1           37s        164m
production-mgmt-mymgmtsite123-postgres-rmdata-cemg       1/1           20s        14m

$ oc get roles -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get rolebindings -n apic -l vendor=crunchydata
No resources found in apic namespace.

$ oc get secrets -n apic -l vendor=crunchydata
NAME                                                  TYPE     DATA   AGE
production-mgmt-7bc754c7-postgres-backrest-repo-config        Opaque   9      6d23h
production-mgmt-7bc754c7-postgres-postgres-secret             Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-primaryuser-secret          Opaque   2      6d23h
production-mgmt-7bc754c7-postgres-testuser-secret             Opaque   2      6d23h
production-mgmt-8aac6b4e-postgres-backrest-repo-config        Opaque   9      6d22h
production-mgmt-8aac6b4e-postgres-postgres-secret             Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-primaryuser-secret          Opaque   2      6d22h
production-mgmt-8aac6b4e-postgres-testuser-secret             Opaque   2      6d22h
production-mgmt-mymgmtsite123-postgres-backrest-repo-config   Opaque   9      2d21h
production-mgmt-mymgmtsite123-postgres-postgres-secret        Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-primaryuser-secret     Opaque   2      2d21h
production-mgmt-mymgmtsite123-postgres-testuser-secret        Opaque   2      2d21h

$ oc get cm -n apic -l vendor=crunchydata
NAME                                                DATA   AGE
production-mgmt-mymgmtsite123-postgres-config       0      2d21h
production-mgmt-mymgmtsite123-postgres-leader       0      2d21h
production-mgmt-mymgmtsite123-postgres-pgbouncer-cm   2      2d21h
production-mgmt-mymgmtsite123-postgres-pgha-config    5      2d21h
```

Procedure

For each type of postgres resource, check to see if any resources are available and if so, delete the resources:

- pods:

  `oc get pods -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete pods -n <APIC_namespace> -l vendor=crunchydata`

- deployments:

  `oc get deployments -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete deployments -n <APIC_namespace> -l vendor=crunchydata`

- Jobs:

  `oc get jobs -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete jobs -n <APIC_namespace> -l vendor=crunchydata`

- roles:

  `oc get roles -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete roles -n <APIC_namespace> -l vendor=crunchydata`

- rolebindings:

  `oc get rolebindings -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete rolebindings -n <APIC_namespace> -l vendor=crunchydata`

- secrets:

  `oc get secrets -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete secrets -n <APIC_namespace> -l vendor=crunchydata`

- cm:

  `oc get cm -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete cm -n <APIC_namespace> -l vendor=crunchydata`

- pgtasks

  `oc get pgtasks.crunchydata.com -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete pgtasks.crunchydata.com -n <APIC_namespace>`

- pgreplica:

  `oc get pgreplica -n <APIC_namespace> -l vendor=crunchydata`

  `oc delete pgreplica -n <APIC_namespace> -l vendor=crunchydata`

VMware

Use the instructions in this section to install, upgrade, and maintain API Connect on VMware.

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

- **Planning your deployment**
  Before you install API Connect, understand deployment options and requirements for endpoints, load balancing, certificates, and firewalls.
- **Installing API Connect**
  Use these instructions to install or upgrade a deployment of API Connect on VMware.
- **Upgrading API Connect**
  Upgrade API Connect on VMware to a newer version.
- **Maintaining API Connect**
  You can use utilities to complete maintenance tasks such as backup, restore, and certificate management in a VMware environment.

Planning your deployment

Before you install API Connect, understand deployment options and requirements for endpoints, load balancing, certificates, and firewalls.

- **Prerequisite skills for administering API Connect on VMware**
  The background knowledge and skills that are required to install and manage an API Connect VMware deployment.
- **Deployment overview for endpoints**
  When deploying API Connect, you will create one or more endpoints for the subsystems and then configure certificates or mutual TLS for most endpoints.
- **Planning your deployment topology and profiles**
  When you install on VMware, API Connect provides a choice of deployment configurations for capacity and high availability (HA). These configurations are named 'deployment profiles' and there are several pre-defined profiles to choose from.
- **Enable JWT instead of mTLS**
  Use JWT to authenticate communication between API Connect subsystems, instead of mTLS.
- **Firewall requirements on VMware**
  Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on VMware.
- **Load balancer configuration in a VMware deployment**
  When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect OVA deployment.
- **Multiple data center deployment strategies**
  Strategies for deploying API Connect on VMware across multiple data centers.
- **Planning your analytics deployment**
  Plan your API Connect analytics deployment by reviewing options for topology, data modifications, and storage.
- **API Connect TLS certificate best practices**
  Best practices for configuring and managing TLS certificates in an API Connect deployment on VMware.

Prerequisite skills for administering API Connect on VMware

The background knowledge and skills that are required to install and manage an API Connect VMware deployment.

The following list provides an idea of the concepts and technologies that the API Connect administrator should be familiar with:

- VMware: Deploying OVA files and taking VM snapshots.
- Linux: File permissions, editing files with the vi editor, redirecting command output to file.
- Network concepts: FQDNs, ports, firewalls.
- TLS: CAs, certificate chains, mTLS.

The examples in this list do not include all background knowledge that is required, but are intended to give a general idea of the skills that are needed to plan, install, and maintain an API Connect deployment.

Deployment overview for endpoints

When deploying API Connect, you will create one or more endpoints for the subsystems and then configure certificates or mutual TLS for most endpoints.

Configuring endpoints

Subsystem	Endpoints	Description	Certificates	
Management	cloud-admin-ui	Configured using the **apicup** command. Endpoint on the management server for communication with the Cloud Manager user interface.	cloud-admin-ui	
	api-manager-ui	Configured using the **apicup** command. API Manager URL endpoint on the management server for communication with the API Manager user interface.	api-manager-ui	
	consumer-api	Configured using the **apicup** command. Platform REST API endpoint for running consumer APIs on the management server.	consumer-api	

Subsystem	Endpoints	Description	Certificates	
	platform-api	Configured using the **apicup** command. Platform REST API endpoint for running admin and provider APIs on the management server.	platform-api	
	hub	Automated Testing Behavior UI and API endpoint. External Frontend/Ingress, port 443	hub-endpoint	
	turnstile	Automated Testing Behavior UI and API endpoint. External Frontend/Ingress, port 443	turnstile-endpoint	
Portal	portal-admin	Configured using the **apicup** command. Corresponds to management endpoint entered in Cloud Manager. Requires a TLS profile configured with mutual TLS.	mutual TLS	
	portal-www	Configured using the **apicup** command. Portal Web site URL entered in the Cloud Manager UI. Used publicly to access Portal.	portal-www-ingress	
Analytics	analytics-client	This is a legacy certificate, it is not used from v10.0.5 onwards.	mutual TLS	
	analytics-ingestion	Configured using the **apicup** command. The analytics-ingestion endpoint must be entered in the Cloud Manager UI when registering the analytics service. It is also used by the gateway service to push data to the analytics service. Requires a TLS profile configured with mutual TLS.	mutual TLS	
Gateway	apic-gw-service	Configured using the **apicup** command. This is the endpoint the gateway uses for network communication. Enter this endpoint as the management endpoint entered in Cloud Manager UI.	apic-gw-service-ingress	
	api-gateway	Configured using the **apicup** command. This is the endpoint the gateway uses for API traffic. Enter this endpoint as the API Invocation Endpoint in Cloud Manager UI.	api-gateway-ingress	

The endpoints are configured by the installation Operator. They are set for each subsystem. Endpoints are also entered when configuring the Topology for the Gateway, Portal, and Analytics subsystems in Cloud Manager.

For instructions on configuring endpoints and installing into an OVA environment, see Installing API Connect.

Configuring certificates

All TLS certificates are automatically generated, but you can customize your user-facing certificates, see API Connect TLS certificate best practices.

Configuring mutual TLS

Mutual TLS is used for communication between API Connect subsystems. Configure inter-subsystem communication in your TLS profiles in the Cloud Manager UI. See Creating a TLS Server Profile.

Configuring a proxy

If a Developer Portal is deployed externally to the management server zone, it does not have access to the consumer and product APIs. You need to configure a proxy to enable communication. For more information, see Configuring a proxy.

Planning your deployment topology and profiles

When you install on VMware, API Connect provides a choice of deployment configurations for capacity and high availability (HA). These configurations are named ‘deployment profiles’ and there are several pre-defined profiles to choose from.

High availability versus disaster recovery

High availability focuses on minimizing the loss of service that is experienced by users during a hardware or software failure. Disaster recovery (DR) focuses on the procedures for recovering a system after a catastrophic hardware, software, or operator failure. Consider the following two metrics:

Recovery Time Objective (RTO)
: The RTO is the time that it is acceptable for a system to be unavailable after a disaster.

Recovery Point Objective (RPO)
: DR solutions are usually based on restoring from a backup. This means that the system can only be recovered to the state it was in when the last backup was taken, and not to the state it was in at the instant the disaster occurred. The RPO measures how far back in time the recovery point is, and therefore how much new data is lost. An RPO of zero would assert that no data is lost, but such a solution is often a compromise against the cost and performance of the system.

To achieve high availability in your API Connect VMware deployment, the three replica deployment profiles should be used, where each API Connect component is distributed across three virtual machines (VMs). However, this topology requires a low network latency between the VMs that is often unachievable between geographically separated data centers. To overcome this limitation, API Connect provides a two data center deployment solution that has both a low Recovery Time Objective (RTO), and a low Recovery Point Objective (RPO).

Deployment options

API Connect is composed of three components: Management, Portal, and Analytics, and the DataPower Gateway supporting program, see API Connect components. Each of these components includes a set of deployment profiles to cover different capacity and availability requirements.
Note: Throughout this document, the Management, Portal, and Analytics components, and the Gateway supporting program are also referred to as "subsystems".
The naming convention of the deployment profiles is `nAxcB.mC` where:

- **nA** indicates the number (A) of VMs (nodes) that are deployed to support the profile. The term "node" is used to refer to a VM instance (n1 means one node, n3 means three nodes).
- **cB** indicates the minimum number of virtual cores (B) required on the VM to deploy the profile.
- **mC** indicates the minimum amount of memory (C) in GB required by the profile on the VM.

API Connect components have two profile types: one replica and three replica:

- The one replica profile is a single node profile and is also known as the `n1` profile. It consists of a single VM for each component. This deployment type is recommended for development or testing where high availability is not a requirement.
- The three replica profiles consist of a cluster of three VMs for each component. This profile is also referred to as the `n3` profile. Each VM runs a replica of the component's microservices, and if a VM fails, the replicas on the other two VMs continue serving requests.

The available profiles for the Management, Portal, and Analytics components are:

- **Management**
  `n1xc4.m16`, `n3xc4.m16`.
- **Portal**
  `n1xc2.m8`, `n1xc4.m16`, `n1xc8.m16`, `n3xc3.m8`, `n3xc4.m8`, `n3xc8.m16`.
- **Analytics**
  `n1xc2.m16`, `n1xc4.m32`, `n1xc6.m48`, `n3xc4.m16`, `n3xc4.m32`, `n3xc6.m48`, `n3xc8.m64`.

For information about gateway CPU and memory, refer to *Deploying the OVA on a VMware hypervisor* in the appropriate version of the DataPower documentation.

Additional points:

- Deployment profiles do not have to be the same for each component. For example, you can have a three replica deployment of the Management component that is configured with a one replica deployment of the Portal component.
- It is possible to change the deployment profile of an existing installation, but all endpoints in your deployment must remain the same. This means that you should use a load-balancer or logical DNS records when you do the initial one replica deployment. For example, if you installed a one replica deployment and used the portal VM hostname as the portal management registration endpoint, this endpoint cannot change when you move it to a three replica deployment. For more information, see Changing deployment profiles on VMware. If you want to change the deployment profile and the endpoints, you can use the form factor migration procedure: Migrating from v10 to v10 on a different form factor.
- Multiple Portal, Analytics, and Gateway component instances can be added with either configuration. For example, you can have a one replica Management component that is configured with two Portal services: a three replica Portal component, and a one replica Portal component.

Two data center disaster recovery

API Connect deployments can operate across two data centers in an active/warm-standby configuration. Each data center has a complete API Connect installation. The active data center processes all user operations. The warm-standby data center does not process user operations but maintains its management and portal component databases in synchronization with the active data center. If a failure occurs at the active data center, the warm-standby data center can become active with a manual failover operation. For more information on the two data center disaster recovery configuration, see: Two data center warm-standby deployment strategy on VMware.

Enable JWT instead of mTLS

Use JWT to authenticate communication between API Connect subsystems, instead of mTLS.

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.
Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on load-balancers.
The following communication flows can be secured with JWT:

- Management initiated communication to portal, analytics, and gateway subsystems.
- Analytics ingestion: The flow of API event records from the gateway to the analytics subsystem.
  Note: If the gateway is under a high transaction load and sending many API events to the analytics subsystem, the enablement of JWT might impact the performance of your analytics subsystem. It is recommended to run performance tests under your expected analytics load before you decide to use JWT for the analytics ingestion path.

With JWT enabled, the portal, gateway, and analytics subsystems verify the JSON Web Token (JWT) sent from the management subsystem when it initiates communication with them. The analytics subsystem verifies the JWT sent from the gateway with incoming API event data. The subsystems that receive the token contact a JSON Web Key Set (JWKS) URL to verify it. The JWKS URL is hosted by the management subsystem, in a subpath of the management subsystem's platform REST API.

You can configure JWT instead of mTLS during installation, as documented in Installing API Connect, or you can configure JWT after installation: Use JWT security instead of mTLS between subsystems.

If you disable mTLS, you must enable JWT. It is not possible to configure API Connect with both mTLS and JWT disabled.

Firewall requirements on VMware

Diagram for port configuration, and list of active ports, for an IBM® API Connect deployment on VMware.

Required Ports between zones

The following network diagram example helps to explain which ports must be configured in an API Connect network. Specific ports must be configured to enable the communication between the various zones, both public and private, in a network.

The ports specified in the diagram are default ports. Check your deployment to understand which communication, if any, is configured to use non-default ports.

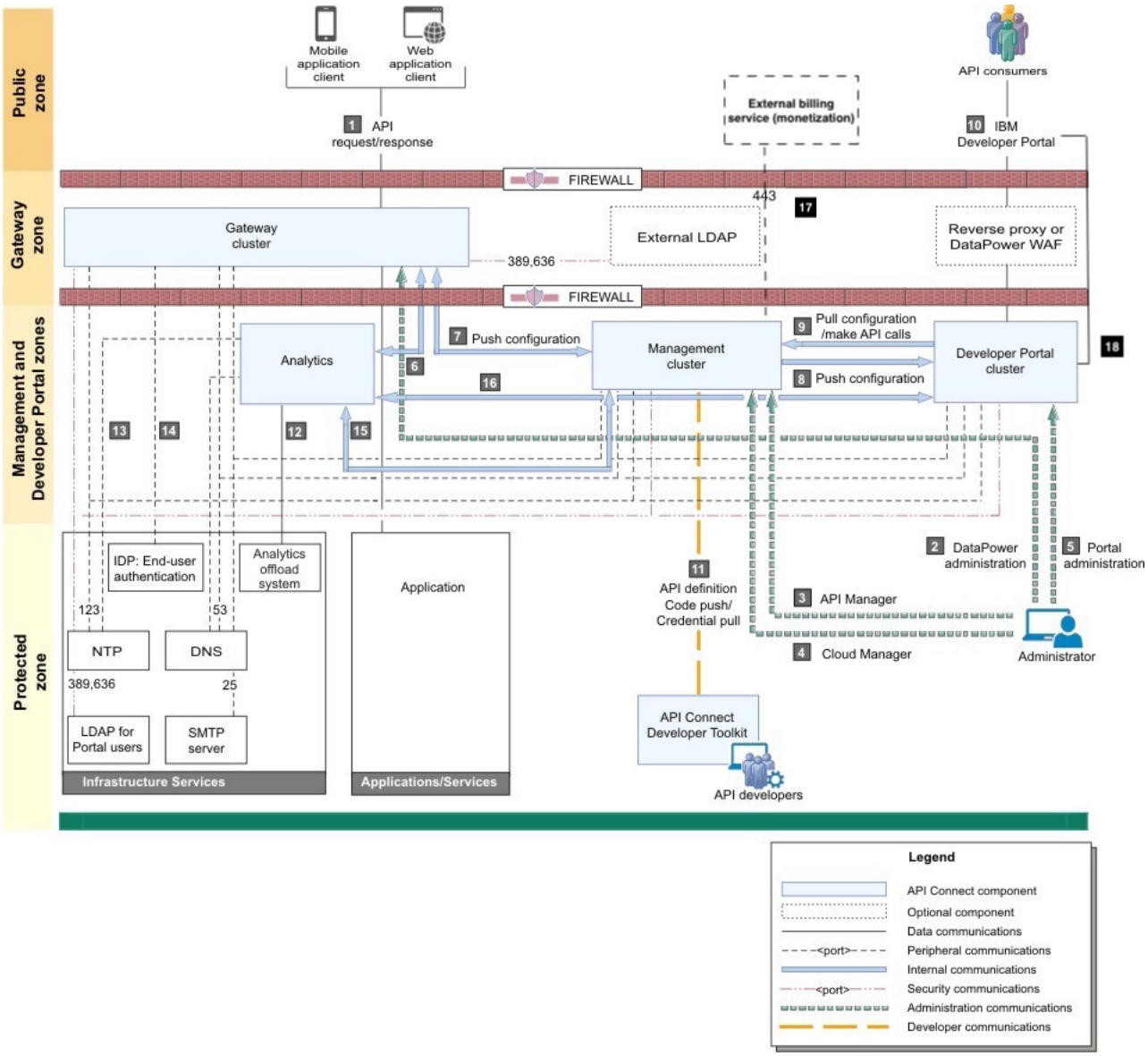


Table 1. Key for the network diagram example.
The following table lists the port numbers with a usage description.

	Usage description	Default port number
1	API request/response – Users invoking the provided APIs.	443 HTTPS from Public zone to Gateway zone
2	DataPower® administration – Internal operators who are managing the Gateway servers.	22 SSH, 9090 HTTPS from Protected zone to Gateway zone
3	API Manager – Internal business users who are defining and monitoring APIs.	443 HTTPS from Protected zone to Management zone
4	Cloud Manager – Internal operators who are administering the Cloud.	22 SSH, 443 HTTPS from Protected zone to Management zone
5	Developer Portal administration – Internal operators who are managing the Portal servers.	22 SSH, 443 HTTPS from Protected zone to Management zone
6	Gateway servers post traffic to Analytics service.	443 HTTPS from Gateway servers to Analytics service
7	Push configuration – Management servers communicate bi-directionally with Gateway servers.	3000 and 443 HTTPS Management servers to and from Gateway servers for webhook delivery. Port 3000 is the default port for apic-gw-service, and this is the port Management uses to communicate with Gateway. Port 443 is the default port for the platform-api endpoint, which apic-gw-service uses to communicate with Management.
8	Push configuration/webhooks – Management servers push configuration and webhooks to the Developer Portal.	443 HTTPS Management servers to Developer Portal servers for webhook delivery
9	Pull configuration/make API calls – Developer Portal servers pull configuration and call REST APIs.	443 HTTPS from Developer Portal servers to Management servers within Management zone
10	Developer Portal – External developers who are accessing the Developer Portal.	443 HTTPS from Public zone to Developer Portal management zone. The reverse proxy/DataPower WAF for incoming web traffic to the Developer Portal cluster must be a transparent proxy - no modification of the portal URL, port, host name or path is allowed.
11	Push API definition to Management server. Pick up credential for microservice code push.	443 HTTPS from Protected zone to Management zone

	Usage description	Default port number
12	Analytics offload	Port will depend on type of plugin and protocol used for the offload. Some possible protocols are: HTTP, HTTPS, TCP, UDP, KAFKA
13	Analytics accesses NTP	Standard NTP
14	Analytics access DNS	Standard DNS
15	Management service queries Analytics service	443 HTTPS within Management zone
16	The Portal service invokes an API (GET) on the Analytics service to retrieve data.	443 HTTPS within Management zone
17	External billing service – Management cluster connecting to external billing service (when configured for billing). If you are using Stripe as your external billing service, you must enable connections with the Stripe API at: https://stripe.com/files/ips/ips_api.json. You can also view the Stripe IP addresses at the following URL: https://stripe.com/docs/ips.	443 HTTPS from Management zone to Public zone
18	Developer Portal cluster must be able to access its own site endpoints.	443 HTTPS from Portal zone to Public zone

Firewall port requirements on VMware

The following tables lists ports that must be open in both cluster and non-clustered deployments.

Note that clustered VMware environment deployments require additional ports. See Firewall enabled ports for clustered OVA deployments

Table 2. Firewall port requirements common to all subsystems

Subsystem	Ports and description
Ports that must be open on all API Connect subsystems	The following ports must be open on the Management Server, Analytics, and Developer Portal subsystems, whether in a cluster or not. • 53 (outbound) DNS • 123 (outbound) NTP • 179 (inbound and outbound) BIRD routing daemon - VMware environments (OVA) deployments. Used for communication over TCP between servers either within the same subsystem or across different subsystems. • 443 (inbound and outbound) Used by all subsystems for communication with other subsystems • 2020 (inbound) status request. Used for communication over TCP between servers within the same subsystem. • 9177 (inbound and outbound) Member list (group membership) for API Connect`apic` daemon communication. Used for communication over both TCP and UDP, between servers within the same subsystem. • 9178 (inbound and outbound) API server, for API Connect `apic` daemon communication. Used for communication over TCP from between servers, both within the same subsystem and across different subsystems. Also used for communication between the `apicup` configuration utility and servers (this is the only port used by `apicup`).

Each subsystem uses ports in addition to the ports in Table 2. See the following table.

Table 3. Additional firewall port requirements for each subsystem

Subsystem	Ports and description
Management Service	Management Service uses the ports in Table 2 plus: • 22 remote backup server port (inbound, configurable) If backups are configured to use another port, ensure that the port is open. See Backing up the management subsystem in VMware environments. Note: this port does not have to be open between the management server and the portal server. • SMTP server port (outbound), typically 25 • 161 (inbound) SNMP. Over UDP. • LDAP server port (if using LDAP user registry), typically 389 (outbound) • In a clustered deployment, the Management Services uses additional ports. See Firewall enabled ports for clustered OVA deployments. The Automated test behavior application must be able to invoke (outbound) any port that a test system might present an API on. It can be bound to a particular port (if it is the same in all environments) or to a range of ports (to cover all of the ports it might invoke).
Developer Portal	The Developer Portal uses the ports listed in Table 2, plus: • 22 - A remote backup server port (inbound, configurable) If backups are configured to use another port, ensure that the port is open. See Backing up and restoring the Developer Portal. Note: this port does not have to be open between the management server and the portal server. • In a clustered deployment, the Developer Portal uses additional ports. See Firewall enabled ports for clustered OVA deployments.
Analytics	The Analytics subsystem uses the ports listed in Table 2, plus: • 161 for SNMP is required for both non-clustered and clustered deployments. UDP, inbound only. • In a non-clustered deployment, no additional ports are required. • Analytics supports an optional configuration for offload of data. This configuration might require additional outbound ports to be open. • In a clustered deployment, the Analytics subsystem uses additional ports. See Firewall enabled ports for clustered OVA deployments.

Subsystem	Ports and description
Gateway Server	The Gateway Server uses these ports in both non-clustered and clustered deployments: • 161 (inbound) SNMP. UDP protocol. Not needed if SNMP not enabled. • 162 (outbound) SNMP traps • 3000 (inbound) Gateway Service local port (configurable) • 5550 (inbound) XML management port (configurable) • 5554 (inbound) REST management port (if enabled; configurable) • 9022 (inbound) Gateway SSH (if enabled; configurable) • 9090 (inbound) Web GUI console (if enabled; configurable) • 9443 (inbound) Gateway local port (configurable) • In a clustered deployment, the Gateway Server uses additional ports. See Firewall enabled ports for clustered OVA deployments.

Communications inside the Gateway cluster

There are a number of important points to note regarding the communications within the Gateway cluster.

- We advise that you use the same port for all Gateway servers within a cluster.
- Gateway servers communicate with each other to synchronize invocation counts.
- All Gateway servers in a Gateway cluster must be able to reach all of the other Gateway servers in the same Gateway cluster.
- Gateway servers in a Gateway cluster do not directly communicate with Gateway servers in a different Gateway cluster.
- All Gateway servers must be able to reach the management subsystem platform API endpoint, which was configured during the installation of your API Connect environment.

Ethernet interface usage

To separate network traffic, you can use two or more Ethernet interfaces on the DataPower appliance on which a Gateway server is installed. For example, you can use one interface for internal IBM API Connect communications, and another for processing incoming API calls.

Port requirement for Stripe billing service

All Management servers and Developer Portal servers must be able to communicate HTTPS content with the external billing service when billing is configured. If you are using Stripe as your external billing service, you must enable HTTPS communication on port 443 with the Stripe API: https://stripe.com/files/ips/ips_api.json.
Note: API Connect does not support using webhooks with Stripe. You can see a list of the Stripe IP addresses at: https://stripe.com/docs/ips.

- **Firewall enabled ports for clustered OVA deployments**
  In a clustered OVA deployment of API Connect, specific ports must be configured for communication between members of each API Connect subsystem.
- **Load balancer ports for clustered OVA deployments**
  When you deploy a load balancer with a clustered deployment, in a VMware environment, you must ensure that the necessary ports are open.

Related concepts

- Firewall enabled ports for clustered OVA deployments
- Load balancer ports for clustered OVA deployments

Firewall enabled ports for clustered OVA deployments

In a clustered OVA deployment of API Connect, specific ports must be configured for communication between members of each API Connect subsystem.

OVA deployments require the common ports that are listed in Firewall requirements on Kubernetes. When the VMs are clustered, additional ports are used for communication between the members of the subsystems in the cluster.

All ports must be enabled inbound and outbound.

Table 1. Firewall enabled ports for clustered VMware environment deployments

Subsystem	Ports
Ports that must be open between all subsystem VMs	442, 2379, 2380, 6443, 6444, 9099, 10248, 10249, 10250,10251, 10252, 10254, 10256,10257, 10259 These ports must be open between all servers within a given subsystem. For example, from management server to management server, or from portal server to portal server, or from analytics server to analytics server. These ports are not used for communication between subsystems. You might need additional ports for Kubernetes-proxied services. The default range is **30000 – 32767**. Since the ports in use can change dynamically, ensure that the default range is open.
Additional ports that must be open between Management Service VMs	Port 8088 is required if the Automated testing behavior application is deployed for the Management Service VMs.
Additional ports that must be open between Developer Portal VMs	3009, 3010, 3306, 3307, 4443, 4444, 4567, 4568, 30865
Additional ports that must be open between Gateway Service VMs	16380, 16381, 26380, 26381
Additional ports that must be open between Analytics VMs	No additional ports are needed.

These internal ports are not used for communication between VMs. Ensure that they are open on the VM server locally.

Table 2. Internal ports reserved by API Connect

Subsystem	Ports
Reserved local ports on all subsystem VMs	8383, 8686, 8080, 8443, 30000:59999
Management Service VMs	2000, 2001, 2002, 2005, 2006, 2007, 2008, 2022, 3003, 3004, 3006, 3007, 3011, 3023, 4150, 4151, 4171, 4222, 5432, 6222, 7777, 8084, 8222, 8404
Portal Service VMs	3058, 3059, 3060, 3061
Analytics VMs	4443, 5000, 5601, 9200, 9300

Note: The ports should support IP-in-IP (protocol 4), per https://docs.projectcalico.org/getting-started/kubernetes/requirements.

Load balancer ports for clustered OVA deployments

When you deploy a load balancer with a clustered deployment, in a VMware environment, you must ensure that the necessary ports are open.

The load balancer needs port 443 for all API Connect subsystems (Management, Analytics, Developer Portal, DataPower Gateway).

Load balancer configuration in a VMware deployment

When deploying API Connect for High Availability, it is recommended that you configure a cluster with at least three nodes and a load balancer. A sample configuration is provided for placing a load balancer in front of your API Connect OVA deployment.

Before you begin

Note:

- This article does not apply to the IBM® DataPower® Gateway component that is a part of the IBM API Connect solution.
- This article applies to third-party software that IBM does not control. As such, the software may change and this information may become outdated.

About this task

API Connect can be deployed on a single node cluster. In this case the ingress endpoints are host names for which the DNS resolution points to the single IP address of the corresponding node hosting a particular subsystem, and no load balancer is required. For high availability, it is recommended to have at least a three node cluster. With three nodes, the ingress endpoints cannot resolve to a single IP address. A load balancer should be placed in front of an API Connect subsystem to route traffic.

Because it is difficult to add nodes once endpoints are configured, a good practice is to configure a load balancer even for single node deployments. With the load balancer in place, you can easily add nodes when needed. Add the node to the list of servers pointed to by the load balancer to avoid changing the ingress endpoints defined during the installation of API Connect.

To support Mutual TLS communication between the API Connect subsystems, configure the load balancer with **SSL Passthrough** and **Layer 4** load balancing. In order for Mutual TLS to be performed directly by the API Connect subsystems, the load balancer should leave the packets unmodified, as is accomplished by Layer 4. Following is a description of the communication between the endpoints that are configured with Mutual TLS:

- API Manager (with the client certificate portal-client) communicates with the Portal Admin endpoint portal-admin (with the server certificate portal-admin-ingress)
- API Manager (with the client certificate analytics-ingestion-client) communicates with the Analytics Ingestion endpoint analytics-ingestion (with the server certificate analytics-ingestion-ingress)

Note: You can disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see Enable JWT instead of mTLS.

Set endpoints to resolve to the load balancer
: When configuring a load balancer in front of the API Connect subsystems, the ingress endpoints are set to host names that resolve to a load balancer, rather than to the host name of any specific node. For an overview of endpoints, see Deployment overview for endpoints and certificates.

Use this example configuration as a guideline to determine the best way to configure the load balancer for your deployment.

Procedure

- **Appliance deployment**
  In this example configuration, the API Connect Management, Portal, Analytics and Gateway subsystems are deployed as three node clusters in Standard mode. An HAProxy load balancer is used. The load balancer is configured with ssl-passthrough and with upstream selection based on SNI. DNS resolution is configured to resolve the endpoints to the IP address of the load balancer. If a single HAProxy node is used, then all endpoints must resolve to the IP address of the single HAProxy. The following example endpoints require DNS resolution for this example:
  - api-manager-ui.sample.example.com
  - cloud-admin-ui.sample.example.com
  - consumer-api.sample.example.com
  - platform-api.sample.example.com
  - admin.portal.sample.example.com
  - web.portal.sample.example.com
  - analytics.ingestion.sample.example.com
  - api-gateway.sample.example.com
  - apic-gw-service.sample.example.com

  Following is an example HAProxy configuration for one HAProxy node distributing traffic to Management and Portal clusters:

Note:

- When you configure a load balancer in front of a Management subsystem, specify timeouts of at least 240 seconds. Note that large deployments might need larger values.

  The default timeout is typically 50 or 60 seconds, which is not long enough to avoid **409 Conflict** or **504 Gateway Timeout** errors. The **409 Conflict** error can occur when the time needed to complete an operation is sufficiently long that a second request gets issued.

  For example, to specify 240 seconds when using HAProxy as a load balancer, set **timeout client** and **timeout server** to **240000**.

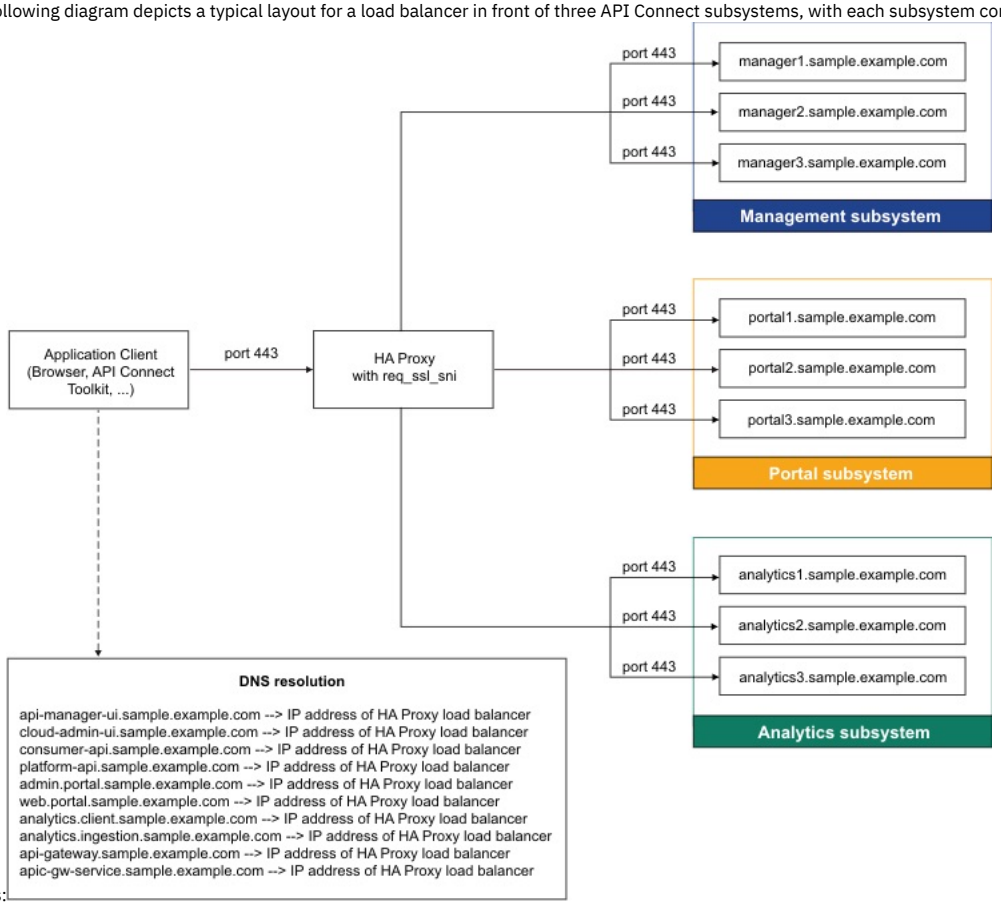
- Ensure that the load balancer is configured to support the cipher **TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)**.
  When the Developer Portal, as a client, calls back to platform-api-endpoint or consumer-api-endpoint of the management subsystem, it sends only a single cipher in the Client Hello. The cipher is **TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384 (0xc030)**. The SSL Handshake will fail if this cipher is not configured (supported) on the client's load balancer.

```
# This sample HAProxy configuration file configures one HAProxy node to distribute traffic to
# Management, Portal, Analytics, and Gateway clusters. Another option is to configure one HAproxy
# node per cluster.

global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:
RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3

defaults
        log     global
        mode    http
        option     httplog
        option     dontlognull
       timeout connect 5000
       timeout client  240000
       timeout server  240000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

######## MANAGEMENT CONFIGURATION ########
frontend fe_management
    mode tcp
    option tcplog
    #
    # Map to the hostname and TCP port for the Management load balancer.
    # In this example, the hostname for the load balancer is ubuntu.sample.example.com.
    #
    bind ubuntu.sample.example.com:443
    tcp-request inspect-delay 5s
    tcp-request content accept if { req_ssl_hello_type 1 }

       #
       # The value for the Management endpoints as defined in the apiconnect-up.yml
       # file using the apicup installer. In this example, the endpoints are api-manager-ui.sample.example.com,
       # cloud-admin-ui.sample.example.com, consumer-api.sample.example.com, and
       # platform-api.sample.example.com. Standard SNI structure specifies
       # whether the INCOMING request is for api-manager or cloud-admin or for consumer-api or platform-api
       # then use "be_management".
       #
    use_backend be_management if
    { req_ssl_sni -i api-manager-ui.sample.example.com OR req_ssl_sni -i cloud-admin-ui.sample.example.com }
    use_backend be_management if
    { req_ssl_sni -i consumer-api.sample.example.com OR req_ssl_sni -i platform-api.sample.example.com }

        #
        # be_management is defined to point management traffic to the cluster
        # containing three management nodes
        #

backend be_management
    mode tcp
    option tcplog
    balance roundrobin
    option ssl-hello-chk

    #
```

```
    # One entry per Management node in the cluster.
    # Hostname and TCP Port for each Management node.
    #
    server management0 manager1.sample.example.com:443 check
    server management1 manager2.sample.example.com:443 check
    server management2 manager3.sample.example.com:443 check


######## PORTAL CONFIGURATION ########
frontend fe_portal
    mode tcp
    option tcplog
    #
    # The Hostname and TCP Port for the Portal Load balancer
    #
    bind ubuntu.sample.example.com:443
    tcp-request inspect-delay 5s
    tcp-request content accept if { req_ssl_hello_type 1 }

    #
    # The value for both of the Portal subsystem endpoints as defined in the apiconnect-up.yml file
    #
    use_backend be_portal if
{ req_ssl_sni -i admin.portal.sample.example.com OR req_ssl_sni -i web.portal.sample.example.com }


backend be_portal
    mode tcp
    option tcplog
    balance roundrobin
    option ssl-hello-chk

    #
    # One entry per Portal node.
    # Hostname and TCP Port for the Portal node.
    #
    server portal0 portal1.sample.example.com:443 check
    server portal1 portal2.sample.example.com:443 check
    server portal2 portal3.sample.example.com:443 check

######## ANALYTICS CONFIGURATION ########
frontend fe_analytics
    mode tcp
    option tcplog
    #
    # The Hostname and TCP Port for the Analytics Load balancer
    #
    bind ubuntu.sample.example.com:443
    tcp-request inspect-delay 5s
    tcp-request content accept if { req_ssl_hello_type 1 }

    #
    # The value for both of the Analytics subsystem endpoints as defined in the apiconnect-up.yml file
    #
    use_backend be_analytics if
{ req_ssl_sni -i analytics.ingestion.sample.example.com }


backend be_analytics
    mode tcp
    option tcplog
    balance roundrobin
    option ssl-hello-chk

    #
    # One entry per Analytics node.
    # Hostname and TCP Port for the Analytics node.
    #
    server analytics0 analytics1.sample.example.com:443 check
    server analytics1 analytics2.sample.example.com:443 check
    server analytics2 analytics3.sample.example.com:443 check

######## GATEWAY CONFIGURATION ########
frontend fe_gateway
    mode tcp
    option tcplog
    #
    # The Hostname and TCP Port for the Gateway Load balancer
    #
    bind ubuntu.sample.example.com:443
    tcp-request inspect-delay 5s
    tcp-request content accept if { req_ssl_hello_type 1 }

    #
    # The values for the Gateway subsystem endpoints as defined in the apiconnect-up.yml file.
    #
    use_backend be_gateway if
{ req_ssl_sni -i api-gateway.sample.example.com OR req_ssl_sni -i apic-gw-service.sample.example.com }


backend be_gateway
    mode tcp
    option tcplog
    balance roundrobin
    option ssl-hello-chk
```

```
    #
    # One entry per Gateway node.
    # Hostname and TCP Port for the Gateway node.
    #
    server gateway0 gateway1.sample.example.com:443 check
    server gateway1 gateway2.sample.example.com:443 check
    server gateway2 gateway3.sample.example.com:443 check
```

- The following diagram depicts a typical layout for a load balancer in front of three API Connect subsystems, with each subsystem containing three Appliance/OVA nodes:



Multiple data center deployment strategies

Strategies for deploying API Connect on VMware across multiple data centers.

If you want to deploy API Connect across multiple data centers, then several options are available and are detailed in this topic.

Note:
You can install the subsystems that make up an API Connect deployment in different data centers, but you must not split a single API Connect subsystem across data centers. For example, if you are deploying a three replica analytics subsystem, then all analytics subsystem VMs (replicas) must run in the same data center.

There is an exception for the gateway service: If TMS is required and an external OAuth provider cannot be used, then the gateways can be located in different data centers if network latency between data centers is less than 30 ms.

Remote gateway deployment

To provide high availability for your published APIs, you can deploy one or more gateway services in different data centers, and register them with the management subsystem in your main data center. If your main data center has an outage, then your remote gateways continue to process calls to your published APIs.

Remote gateway deployment is the simplest way to take advantage of multiple data centers. The main disadvantage is that you cannot update any published APIs during the main data center outage (because your management subsystem is in the main data center).

Cold-standby deployment

In a cold-standby deployment, API Connect is installed and running in your main data center. In your cold-standby data center, API Connect is ready to be installed and has access to the API Connect backup files from your main data center.

If your main data center has an outage, then you install API Connect in your cold-standby data center, and restore your most recent subsystem database backups to your cold-standby deployment.

Requirements for cold-standby configuration:

- The API Connect deployment in the cold-standby data center must use the same deployment profile as the main data center.

- The API Connect network endpoints that you define in your main data center must be valid in your cold-standby data center.
  For example, if the network endpoint for your Cloud Manager UI is **`cloud-admin.apicinstall.example.com`**, this endpoint must route correctly to the management subsystem you install in your cold-standby data center.

  Every network endpoint that you configure during API Connect installation and configuration in your main data center must be valid in your cold-standby data center.

- Ensure that subsystem backups are regularly copied from your main data center to your cold-standby. Backups must include both your project directory and your subsystem database backups. See Backing up and restoring on VMware.

The steps for failing over to your cold-standby data center are:

1. Install API Connect in your cold-standby data center.
2. Restore your API Connect subsystem database backups.
3. Update network routing so that user traffic is directed to your cold-standby data center.

Steps 1 and 2 are essentially the same as the disaster recovery steps to restore API Connect, see Recovering from a disaster.

The advantage of the cold-standby deployment over the remote gateway deployment is that your full API Connect deployment is available in the cold-standby site. Management and portal subsystem users can update APIs and view analytics data while your main site is unavailable.

Warm-standby deployment

In the warm-standby deployment, the management and portal subsystems continually replicate their databases to a warm-standby API Connect deployment in a separate data center.

The advantage of the warm-standby deployment over cold-standby is that the failover process is faster, and the management and portal databases in the warm-standby site are fully up to date. With the cold-standby strategy, your management and portal databases are only as recent as the last backup you took from your main site.

The disadvantage of the warm-standby deployment is that it adds extra complexity to the API Connect installation process, and all ongoing maintenance procedures such as upgrade. Use the warm-standby deployment strategy only when the following are essential requirements:

- Minimal management and portal subsystem downtime.
- No management and portal data loss, when restoration of data from the last backup that was taken of your active site is not sufficient.

Requirements for warm-standby configuration:

- The API Connect deployment in the warm-standby data center must use the same deployment profile as the main data center.
- The API Connect network endpoints that you define in your main data center must be valid in your warm-standby data center.
  For example, if the network endpoint for your Cloud Manager UI is **`cloud-admin.apicinstall.example.com`**, this endpoint must route correctly to the management subsystem you install in your warm-standby data center.

  Every network endpoint that you configure during API Connect installation and configuration in your main data center must be valid in your warm-standby data center.

The warm-standby deployment configuration is referred to in this documentation as two data center disaster recovery (2DCDR). For more information about 2DCDR, see Two data center warm-standby deployment strategy on VMware.

- **Two data center warm-standby deployment strategy on VMware**
  An overview of the two data center disaster recovery deployment strategy in API Connect.

Two data center warm-standby deployment strategy on VMware

An overview of the two data center disaster recovery deployment strategy in API Connect.

The two data center disaster recovery warm-standby deployment configuration (referred to as 2DCDR) provides continuous replication of the management and portal subsystem databases to a warm-standby deployment in another data center.
Important: The 2DCDR deployment adds complexity to the API Connect installation procedures, and all subsequent maintenance procedures such as backup, restore, and upgrade. Before you decide to use the 2DCDR deployment strategy, review the Multiple data center deployment strategies to see whether any of the alternatives are better suited to your requirements.

Key points of the two data center disaster recovery (DR) solution

- Two data center DR is an active/warm-standby deployment for the API Manager and Developer Portal services, and must use manual failover.
- Two DataPower® Gateway subsystems must be deployed to provide high availability for the gateway service. However, this scenario doesn't provide high availability for the analytics service.
- If high availability is required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.
- Data consistency is prioritized over data availability.
- Latency between the two data centers must be less than 80 ms.
- Replication of the API Manager is asynchronous, so it is possible that the most recent updates do not transfer to the warm-standby data center if there is an active data center failure.
- Replication of the Developer Portal is synchronous, and therefore the latency is limited to 80 ms or less.
- The API Manager and the Developer Portal services in the two data centers must use the same deployment profile.
- It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a two data center disaster recovery configuration.

Deployment architecture

A two data center deployment model is optimized for data consistency ahead of data availability, and must use manual failover when a fault occurs. For high availability of Management and Portal subsystems ensure that you use a three replica deployment profile. For more information on deployment profiles, see: Planning your deployment topology and profiles.

To achieve high availability for the DataPower Gateway, you must deploy two gateway subsystems. One subsystem in the active data center, and a separate subsystem in the warm-standby data center. Publish all Products and APIs to both gateway subsystems. The gateway subsystems are independent, and so are insulated if an issue occurs in one of them. A global dynamic router can then be used to route traffic to one gateway subsystem or the other. If high availability is also required for the analytics service, two analytics subsystems must be configured, one per gateway subsystem, but with this configuration Developer Portal analytics isn't possible.

Dynamic routing

Note: A dynamic routing device, such as a load balancer, is required to route traffic to either data center. However, neither this device nor its configuration is part of the API Connect offering. Contact IBM Services if you require assistance with configuring a dynamic router.
The dynamic router configuration must handle the traffic between the subsystems and between the data centers. For example, the consumer API calls from the Developer Portal to API Manager must transfer through a dynamic router so that the Developer Portal can use a single endpoint regardless of which data center API Manager is active in. The same is needed for calls to the Platform and Admin APIs from the other subsystems, as well as for incoming UI traffic for the Developer Portal UI, Cloud Manager UI, and API Manager UI.

The dynamic router must support SSL passthrough, so that it routes the Mutual TLS (mTLS) connections between API Manager and the Developer Portal, and between API Manager and the DataPower Gateway. The router should not do TLS termination, it should do layer 4 based routing by using SNI.
Note: You can disable mTLS and use JWT instead, which allows the load-balancers to do TLS termination. For more information, see Enable JWT instead of mTLS.

Service status

When a failure occurs, it is common practice to display an interstitial system outage web page. The dynamic router can be configured to display this web page when there is a failure, while the manual failover to the warm-standby data center is taking place.

Deployment profiles

Both one replica and three replica deployment profiles can be used with two data center DR, but they must be the same at each data center. For more information about the deployment profiles, see Requirements for initial deployment on VMware.

For more information about a two data center DR deployment, see the following topics:

VMware

- Installing a two data center deployment
- Maintaining a two data center deployment (including information about normal operation data flows)
  - How to failover a two data center deployment V10.0.6.0
  - Recovering from a failover of a two data center deployment V10.0.6.0
  - Backup and restore considerations for a two data center deployment
  - Upgrading a two data center deployment

Planning your analytics deployment

Plan your API Connect analytics deployment by reviewing options for topology, data modifications, and storage.

Deployment profile

The first decision that you must make is which **`deployment-profile`** you want to use. The available deployment profiles for analytics are: **`n1xc2.m16`**, **`n1xc4.m32`**, **`n1xc6.m48`**, **`n3xc4.m16`**, **`n3xc4.m32`**, **`n3xc6.m48`**, **`n3xc8.m64`**. For more information about deployment profiles, see Deployment profiles.

For a typical production deployment, choose a three replica profile. A one replica profile is only sufficient if you expect a low API transaction rate and do not plan to retain much analytics data.

Storage space

Based on your expected API transaction rate, you can estimate how much storage space your analytics subsystem requires. See Estimating storage requirements.

Firewall requirements

The analytics firewall requirements are documented here: Firewall requirments.

Inter-subsystem communication security

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.
Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on load-balancers.
For more information about configuring JWT for network security, see: Enable JWT security.

Disabling internal storage

If you are offloading your analytics data to a third-party system, you can choose to not store your API event data in your API Connect analytics subsystem by disabling local analytics storage. With local storage disabled, the CPU, memory, and storage requirements of your analytics subsystem are greatly reduced, but you lose the ability to view your analytics data in the API Connect UIs.
Important: It is not possible to enable or disable local storage after installation.

API Connect TLS certificate best practices

Best practices for configuring and managing TLS certificates in an API Connect deployment on VMware.

API Connect generates TLS certificates during installation that are used to secure all network communication to, from, and within your API Connect deployment.

The TLS server certificates that secure the user-facing endpoints of your API Connect deployment can be replaced with your own organization's certificates. For example, if you want portal users to see your organization's certificate instead of the auto-generated API Connect certificate, you can update the certificate that secures the **portal-www** endpoint to use your own certificate.

If more than one endpoint is set to the same host, the certificates associated with the endpoints should be set to the same.

The following table lists all the user-facing certificates in API Connect:

Table 1. User-facing endpoint certificates.

Certificate name	Description
portal-www-ingress	The **portal-www-ingress** certificate is a user-facing ingress certificate that is returned to users when they attempt to access developer portal sites. The endpoint is called **portal-www** in the portal subsystem configuration.
api-manager-ui	The **api-manager-ui** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Manager UI. The endpoint in the management subsystem configuration uses the same name: **api-manager-ui**. The hostnames for which the certificate is valid must include the **api-manager-ui** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **serverAuth**.
cloud-admin-ui	The **cloud-admin-ui** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the Cloud Manager UI. The endpoint in the management subsystem configuration uses the same name: **cloud-admin-ui**. The hostnames for which the certificate is valid must include the **cloud-admin-ui** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **serverAuth**.
consumer-api	The **consumer-api** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at organization scope. The endpoint in the management subsystem configuration uses the same name: **consumer-api** The hostnames for which the certificate is valid must include the **consumer-api** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **serverAuth**.
platform-api	The **platform-api** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at cloud scope. The endpoint in the management subsystem configuration uses the same name: **platform-api** The hostnames for which the certificate is valid must include the **platform-api** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **serverAuth**.
hub	Used by the Automated API behavior testing application. User-facing ingress certificate for the **hub-endpoint**.
turnstile	Used by the Automated API behavior testing application. User-facing ingress certificate for the **turnstile-endpoint**.

Note: The API invocation certificate (the certificate that is presented to callers of your APIs) is configured in the Cloud Manager UI when you register your gateway services. For more information about API invocation certificates, see Registering a gateway service and TLS profiles overview.
Note: For more information about the non-user facing certificates used in API Connect, see TLS certificate concepts and operations

- **Customizing your user-facing certificates**
  How to set the user-facing endpoints of API Connect to use your own TLS server certificates.

Customizing your user-facing certificates

How to set the user-facing endpoints of API Connect to use your own TLS server certificates.

Before you begin

Identify the user-facing certificates that you want to customize: User-facing certificates.

Ensure that you have certificate, key, and CA PEM files for each certificate you want to customize. The PEM files must not be password-protected.

For example, to customize the certificate that secures the Cloud Manager UI, you must have the following PEM files: cloud-admin-ui-cert.pem, cloud-admin-ui-key.pem, cloud-admin-ui-ca.pem

User-facing certificates must meet the following requirements:

- Specify Extended Key Usage (EKU) of **serverAuth**.
- Include the Subject Alternative Name (SAN) of the endpoint that they secure.
- If the certificate is signed by an intermediate CA, then the ca.pem file must contain the full chain to root CA.

About this task

You update the user-facing certificates with the **apicup** command.

You can set custom user-facing certificates during installation, or you can update them after installation, replacing the default certificates that are generated.

Note: API Connect verifies certificates when you set them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your custom certificates before they expire.

Procedure

1. If you are updating a user-facing certificate after installation, you must first clear the default certificate that was created:

   ```
   apicup certs set <subsystem name> <certificate name> --clear
   ```

   For example, to clear the certificate that secures the Cloud Manager UI:

   ```
   apicup certs set mgmt cloud-admin-ui --clear
   ```

2. Set the user-facing certificate with the **apicup certs set** command, supplying the PEM files for your new certificate:

   ```
   apicup certs set <subsystem name> <certificate name> <cert file> <key_file> <ca file>
   ```

   For example:

   ```
   apicup certs set mgmt cloud-admin-ui cert.pem key.pem ca.pem
   ```

3. If you are updating the certificate after installation, then apply the update with `apicup subsys install`

   ```
   apicup subsys install <subsystem name>
   ```

4. You can verify that the certificate is updated with **apicup certs list**:

   ```
   apicup certs list <subsystem name>
   ```

   Example output:

   ```
   Common certificates
   ===================

   Name                        Summary                                                                 Validation errors
   ----                        -------                                                                 -----------------
   ...

   Subsystem def-management certificates
   =====================================

   Name                              Summary                                                           Validation errors
   ----                              -------                                                           -----------------
   ...
   cloud-admin-ui                    CN: cloud-admin-ui
                                     SubjectKeyId: 9E:8B:FA:7E:92:70:EC:86:2F:C3:3A:20:E8:13:EA:F4
                                     AuthorityKeyId: E4:AB:09:FF:A9:7D:D3:70:DD:AD:69:C4:A4:34:8D:96
   ...
   ```

   Note: Along with the user-facing certificates, the **apicup certs list** command outputs details of other internal API Connect certificates and secrets that you should not update.

If you see validation errors in **apicup certs list** output, check Certificate validation reference.

If you want to revert to using automatically generated user-facing certificates, then you can clear and regenerate the certificates by following these steps:

1. Clear the certificate:

   ```
   apicup certs set <subsystem name> <certificate name> --clear
   ```

   Repeat for all certificates that you want to clear.
2. Generate new certificates for all the cleared certificates:

   ```
   apicup certs generate <subsystem name>
   ```

3. If you are updating the certificate after installation, then apply the update with `apicup subsys install`:

   ```
   apicup subsys install <subsystem name>
   ```

Installing API Connect

Use these instructions to install or upgrade a deployment of API Connect on VMware.

- **IBM API Connect Version 10 software product compatibility requirements**
  Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.
- **Requirements for initial deployment on VMware**
  Review the requirements and considerations for deploying in a VMware environment.

- **API Connect licenses**
  When you install a new version of API Connect on VMware, you must accept a new license.
- **What's new for v10 installation**
  Summary of installation changes for API Connect v10 on VMware.
- **First steps for deploying in a VMware environment**
  The first steps in deploying in a VMware environment are to obtain the API Connect distribution files and to create a project directory.
- **Verifying the integrity of IBM product files**
  You can verify the integrity of files to ensure that they originated from IBM and are not modified.
- **Deploying the Management virtual appliance**
  You can create a virtual server by deploying the relevant IBM API Connect OVA file on a VMWare virtual server. Create all of the virtual servers that you want to use in your cloud.
- **Deploying the Analytics virtual appliance**
  Define your analytics subsystem, create an ISO file for each analytics node, and deploy the analytics nodes using the analytics OVA file on a VMware virtual server.
- **Deploying the Developer Portal virtual appliance**
  You create a Developer Portal node by deploying the Developer Portal OVA template. After you deploy the Developer Portal OVA template, you can install the Developer Portal.
- **Deploying DataPower Gateway virtual appliance**
  API Connect uses IBM DataPower® Gateway to provide the gateway service.
- **Post-deployment steps**
  When all subsystems are deployed, use the admin account to access the Cloud Manager administrative console and begin configuration. Optionally, take VMware snapshots of the VMs.
- **Installing a two data center deployment**
  How to install a two data center disaster recovery (2DCDR) deployment on VMware.

IBM API Connect Version 10 software product compatibility requirements

Ensure that you install the minimum API Connect operating system requirements. Use the IBM® Software Product Compatibility Reports site to generate a requirements report appropriate for your API Connect version and environment.

Generating a Software Product Compatibility Report

To generate an API Connect requirements report, complete the following steps:

1. Open the Detailed system requirements for a specific product page on the IBM Software Product Compatibility Reports site.
2. Search for the IBM API Connect product.
3. In the Search results list, select IBM API Connect.
4. From the Version list, select the required version.
5. Use the Filters to refine the contents of the requirements report.
6. Click Submit to generate your requirements report.

Important: When reviewing the report to determine your deployment needs, you should treat the stated CPU and memory requirements as reserved VMware capacity. For example, if the system requirements are 4 CPU and 16GB RAM, then those resources should be available at all times from the underlying hardware and should not be coming from an over-committed pool of resources.

Requirements for initial deployment on VMware

Review the requirements and considerations for deploying in a VMware environment.

- Deployment requirements on VMware
- Disk space requirements
- API Connect configuration on VMware
- Passwords and certificates
- Setting and using a hashed default password
- DataPower Gateway for API Connect on VMware

Deployment requirements on VMware

- Ensure you have supported software requirement versions. See IBM API Connect Version 10 software product compatibility requirements
  Attention: API Connect is not supported on a FIPS-enabled environment.
- Do not change the hardware version of the OVA during installation. Do not attempt to use an unsupported version, even if VMware indicates compatibility with other versions. For example, when deploying API Connect, the VMware UI for the APIC OVAs might show information like:

Table 1.

Property	Value
Guest OS	`Ubuntu Linux (64-bit)`
Compatibility	`ESXi 5.5 and later (VM version 10)`
VMware Tools	`Yes`
CPUs	`4`
Memory	`16 GB`

Although the **Compatibility** field shows `ESXi 5.5 and later (VM version 10)`, API Connect supports only the versions listed in IBM API Connect Version 10 software product compatibility requirements. Do not change the VM version of the OVA. Ensure that the **Compatibility** field values are not changed, and remain `ESXi 5.5 and later (VM version 10)`.

Attempts to modify the VMware compatibility may typically result in failure to boot the OVA, as per https://kb.vmware.com/s/article/52683.

- Ensure that your operating system has one of the supported utilities for creating ISOs. The apicup installer uses `mkisofs` on Linux, and `hdiutil` on macOS. For Windows, you need software that creates ISO files using `mkisofs`, such as CDRTools. Verify that the utility you will use is located in a directory that is referenced by the PATH environment setting for your operating system. When creating the ISO, if you encounter the message `Error: unable to create config ISO for host`, verify that you have sufficient permissions to run the command.
- **Important**: Use a single `apicup` project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.

  The original project directory created with `apicup` during the initial product installation (for example, *myProject*) is required to both restore the database and to upgrade your deployment. You cannot restore the database or perform an upgrade without the initial project directory because it contains pertinent information about the cluster. Note that the endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored or upgraded system. A good practice is to back up the original project directory to a location from where it can always be retrieved.

  Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.

- For each subsystem, gather the following networking settings, which you will need to supply during configuration:

Table 2.

Required information	Value for your system
IP address of the server	
Domain of the server	
IP addresses of the name servers	
IP address of the network gateway (not DataPower® gateway) for the server	
Name of the Ethernet interface	
VLAN	

Some virtualization environments require additional information when you create and configure virtual machines. For example, it might be necessary to assign a specific VLAN ID, Resource Pool, or Datastore. Please refer to information provided by your virtualization environment administrators.

Disk space requirements

- For Version 10, the data disk requirement for the management server is 200 GB. The boot disk requires 100 GB, so a minimum of 300 GB is needed for the management subsystem. You specify the data disk size in Deploying the Management subsystem OVA file when you deploy the OVF template in the VMware vSphere Web Client.
- The Developer Portal disk requirements vary with usage scenarios. See Table 1.
- Before you install, review all the minimum hardware requirements listed on the Hardware tab in the Software Product on the Compatibility Reports for API Connect. Follow the instructions in IBM API Connect Version 10 software product compatibility requirements.

API Connect configuration on VMware

- API Connect cannot be deployed on NFS.
- The timezone for API Connect pods is set to UTC. In API Connect deployments on VMware, the operating system timezone is also set to UTC. Do not change the timezone for the pods or the operating system.
- Ensure that the time settings match (within a few seconds) between the machine running `apicup` and the VMware `Host` clock. To verify the VMware host clock setting, see https://kb.vmware.com/s/article/1003736.

  If the clocks are too different, the installation can fail because of invalid certificates due to time discrepancies.

- Deploying multiple similar subsystems in a single namespace is not supported. For example, deploying two management subsystems, `mgmt1`, `mgmt2` in a single namespace where apiconnect operator is installed.
- API Connect requires a dedicated IP range for deployment of the Kubernetes pod and Kubernetes service networks. These IP addresses cannot conflict with IP addresses used by other resources in your deployment, such as SMTP servers or user registries. The default values are 172.16.0.0/16 and 172.17.0.0/16, respectively. If a /16 subnet overlaps with existing IPs on the network, a CIDR as small as /22 is acceptable.
  If the default ranges conflict with other programs, you can modify the API Connect ranges during initial installation. Note that you cannot modify them once an appliance has been deployed. Follow the instructions for your subsystem if you want to modify either of the IP ranges.

- Only static IP addresses that are specified during the `apicup` project configuration before the installation of the OVAs are supported.
- Designated host names must have wildcard aliases or host aliases, which ensures that the different endpoints work together. For example, **.hostname.mycompany.com*.
- Endpoints cannot contain the underscore character `"_"` or uppercase characters.
- Host names that are used to create endpoints cannot contain the underscore character `"_"` or uppercase letters.
- Kubernetes ingress limits the character set for DNS names to not support the underscore character `"_"`. This means you cannot specify underscores in domain names that are used as endpoints. For example, `my_domain.bar.com` and `my.domain_abc.com` are not supported for `<xxx>.<hostname>.<domainname>`, and will cause an error. For example:

  ```
  Invalid value: "my_domain.bar.com": a DNS-1123 subdomain must consist of lowercase alphanumeric characters, '-' or '.',
  and must start and end with an alphanumeric character (e.g. 'example.com', regex used for validation is
   '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')
  ```

- Note that you cannot change host names or DNS names on a running cluster.
- For Analytics, the installer specifies a default value of 16 GB for `es-max-memory-gb`. The minimum value for this setting is 12 GB.
- Decide the *deployment-profile* to use for the installation. See Planning your deployment topology and profiles.
  Use of three replica deployment-profile is supported only on installations with three or more nodes. Installations with less than three nodes must use a one replica deployment-profile.

  Note that a deployment-profile is set for each subsystem type: Management, Analytics, and Developer Portal.

- For additional considerations for configuring clusters, see Configuring API Connect subsystems in a cluster.
  Important: For best performance it is recommended that the network latency between any 2 nodes be as low as possible. Do not configure nodes from the same subsystem cluster across multiple data centers with a high latency network. A high latency network is one that experiences more than 30ms latency between nodes.

- Configure a remote server for logging. Follow these instructions: Configuring remote logging for a VMware deployment. Pods are terminated during an upgrade, and the logs will be lost if not stored remotely.

Passwords and certificates

- When creating configuration files for use in generating an ISO image for VMware, ensure that your working directories are secure. The VMware configuration requires an ISO image that contains a plain text password to be used to unlock the VMware data disk. This means that the API Connect project configuration file apiconnect-up-v10.yml, and the /user-data directory for each host, contain the passwords you specified. This configuration information is used to create the ISO image that you combine with the ..ova distribution files when deploying (unlocking) the VMware data disk.
- You can use **apicup** to specify an **ssh** keyfile that contains a public certificate for using **ssh** to log in to a virtual machine. Logging in through **ssh** is preferred because it is more secure than password-based login.
- Default certificates are generated for each subsystem by the `apicup subsys install` command. If certificates are not explicitly set using the `apicup certs set` command, then default certificates are automatically generated by **apicup**. See API Connect TLS certificate best practices.

Setting and using a hashed default password

During configuration of the Management, Analytics, and Developer Portal subsystems, you create a password to use to log in to the management console for the first time. You must use a password hashing utility to hash the password. You then use **apicup** to assign this hashed password to the subsystem. These configuration steps ensure that the password in not stored in plain text on the data disk.

The syntax of the **apicup** command is:

```
apicup subsys set mgmt default-password='hashed_password'
```

Usage notes:

- The **default-password** is for the **apicadm** user account on the Appliance.
- The password for **apicadm** can be used **only** to log in through the VMware console. You cannot use it to **ssh** into the Appliance as an alternative to using the **ssh-keyfiles**. Interactive login for **ssh** is disabled.
- The **default-password** value configured is only used during initial installation (first boot) of each virtual appliance. Changing the value, then regenerating the ISO, and attaching the new ISO to the virtual appliance does not change the **apicadm** password.
- The **default-password** must be hashed. If it is plain text, you will not be able to log into the Appliance through the VMware console. When you use **apicup** to set or get **default-password**, **apicup** ensures that the hash type of the password is one of the following:
  - MD5
  - SHA1
  - SHA256
  - SHA512
  - BCRYPT
  - MD5-Crypt
- When using **apicup** to set a default password for a subsystem, be aware of syntax differences between operating systems. Windows requires double-quotes. Linux and OSX require single quotes.

Operating system	Command syntax
Linux or OSX	`apicup subsys set mgmt default-password='hashed_password'`
Windows	`apicup subsys set mgmt default-password="hashed_password"`

- You can use the **passwd** command (on the appliance) to change the **apicadm** password.
- When using the VMware Remote console to login to the appliance, be aware that the keyboard layout is English. This can cause a problems with hashed passwords, if you created the ISO on a system with a different keyboard layout and you used special characters or symbols. Passwords are hashed when you set the password that you enter to log into your Management appliance for the first time, and when you create hosts.

DataPower Gateway for API Connect on VMware

- Installation and configuration of DataPower Gateway on an appliance (physical or virtual) is completed after you install the API Connect subsystems. For the gateway service in a VMware environment, use the instructions in Deploying DataPower Gateway virtual appliance.
- Ensure that DataPower Gateway firmware version you plan to install is compatible with the API Connect Management server version.
  Note:
  You can use any combination of API Connect CD releases (10.0.6.x) with DataPower CD releases (DataPower Gateway 10.5.1.x or DataPower API Gateway 10.5.1.x).

  Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

API Connect licenses

When you install a new version of API Connect on VMware, you must accept a new license.

Setting the License ID on VMware

Each license has a unique identifier called a **License ID**. Use **apicup** to accept the license. License acceptance is stored in **apiconnect-up-v10.yml**. Each accepted **License ID** is passed into subsystem CRs.

```
apicup licenses accept <License_ID>
```

For example:

`apicup licenses accept L-KZXM-S7SNCU`

- Use `apicup licenses list` displays licenses available for current version. Also shows all accepted licenses, including prior versions. `apicup licenses list` has a `-o yaml|json` flag to make script based usage easier.
- If an invalid value is passed to `apicup licenses accept`, an error is displayed showing the valid values:

  ```
  Error: Must run `apicup licenses accept ID` with one of [L-GVEN-GFUPVE, L-BYRR-EEASJ3, .....]
  ```

- If you run other `apicup` commands, such as `apicup subsys get <subsystem_name>` from within a project directory, without first accepting a license, those commands will fail. Once you accept the license, the commands will succeed.

You accept the license during First steps for deploying in a VMware environment.

Beginning with API Connect 10.0.6.0, the licenses in Table 1 will be used as the master mod-level license for all Fix Pack and security roll-up releases in this stream. There are no changes to license terms, only updates to open-source notices documented in the notes for each.

Table 1. API Connect 10.0.7 licenses

License ID	Program Name	URL
L-MMBZ-295QZQ	IBM API Connect Enterprise V10.0.7.0	https://ibm.biz/BdMLs3
L-GGQD-G7AYJD	IBM Cloud Pak for Integration - API Calls 2023.4.1	https://ibm.biz/BdMLiR
L-JWFC-XU2KXR	IBM API Connect Enterprise Add-on for IBM Products V10.0.7.0	https://ibm.biz/BdMLsb
L-LHBS-DGEM6J	IBM API Connect Professional V10.0.7.0	https://ibm.biz/BdMLsi

Table 2. API Connect 10.0.6 licenses

License ID	Program Name	URL
L-KZXM-S7SNCU	IBM API Connect Enterprise V10.0.6.0	https://ibm.biz/BdPJsn
L-NRHE-WFD7EY	IBM Cloud Pak for Integration - API Calls 2023.2.1	https://ibm.biz/BdPJse
L-TBPQ-X87AGB	IBM API Connect Enterprise Add-on for IBM Products V10.0.6.0	https://ibm.biz/BdPJsb
L-FNPP-MXMAH5	IBM API Connect Professional V10.0.6.0	https://ibm.biz/BdPJsp

What's new for v10 installation

Summary of installation changes for API Connect v10 on VMware.

- Changes to `apicup` settings:
  - `license-use`. Values changed: `Production` is now `production`. `Nonproduction` is now `nonproduction`.
  - `licenses accept` added for setting Program license. See API Connect licenses
  - `mode` is deprecated and replaced by `deployment-profile`. Each subsystem (Management, Analytics, Portal) has separate values for one replica profiles and three replica profiles.
  - v2018 settings that are deprecated:
    - az-name - no longer used with the Management subsystem
    - cross-az-peers- no longer used with the Management subsystem
    - cassandra-max-memory-gb - no longer used with the Management subsystem
    - es-memory-gb - no longer used with the Analytics subsystem.
    - All cassandra-backup-* settings are no longer used with the Management subsystem. The Cassandra database is deprecated and replaced by Postgres.
      - cassandra-backup-auth-pass
      - cassandra-backup-auth-user
      - cassandra-backup-host
      - cassandra-backup-path
      - cassandra-backup-port
      - cassandra-backup-protocol
      - cassandra-backup-schedule
- The minimum size data disk for the Management system increases from 150 GB in v2018 to 200 GB in v10. When combined with the 100 GB requirement for the root disk (unchanged), you need 300 GB for the Management subsystem.
- Upon reboot, any extra disk that was added to the VM is picked up and added to the secure storage (data disk).

First steps for deploying in a VMware environment

The first steps in deploying in a VMware environment are to obtain the API Connect distribution files and to create a project directory.

Before you begin

- Review Requirements for initial deployment on VMware.
- See What's new for v10 installation
- See also Configuring API Connect subsystems in a cluster.

About this task

In these First Steps, you will use the `apicup` command to create a project directory that contains the configuration file `apiconnect-up-v10.yml`. When you configure the first subsystem, such as the Management Service, the subsystem configuration settings are placed in the file.

When you want to configure a second subsystem, it is important that you reuse the same project directory. This means that you only need to download **apicup** once, and create a project directory once. The configuration settings for the second subsystem are added to the existing **apiconnect-up-v10.yml**. In this way, the second subsystem, such as Developer Portal, can access the configuration information needed to interact with the first subsystem.

After completing these First Steps, follow the links to instructions for configuring your subsystem. You will specify configuration settings, create an ISO of the subsystem configuration, and then use VMware to deploy the distribution file (.ova file) for the subsystem with your ISO. Note that you create a separate ISO for each subsystem.

Installation and configuration of DataPower® Gateway on an appliance (physical or virtual) is completed after you install the API Connect subsystems. For the gateway service in a VMware environment, use the instructions in Deploying DataPower Gateway virtual appliance.

Note:

- For upgrades, do not use the procedures on this page. See Upgrading API Connect.
- Some **apicup** commands have changed from previous versions of API Connect. See What's new for v10 installation.
- When maintaining API Connect, do not use **kubectl exec** commands to access API Connect pods unless advised by IBM.
  Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

Procedure

1. Obtain the API Connect files.
   The following files are used for initial deployment on VMware:

   IBM® API Connect *<version>* Management for VMware
   : Management subsystem files

   IBM® API Connect *<version>* Developer Portal for VMware
   : Developer Portal subsystem files

   IBM® API Connect *<version>* Analytics for VMware
   : Analytics subsystem files

   IBM® API Connect *<version>* Install Assist (apicup) for *<operating_system_type>*
   : The apicup installation utility. Required for all installations on VMware.

   IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
   : Toolkit command line utility. Packaged standalone, or with API Designer or Loopback:
     - IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
     - IBM® API Connect *<version>* Toolkit with Loopback for *<operating_system_type>*
     - IBM® API Connect *<version>* Toolkit Designer with Loopback for *<operating_system_type>*

     Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See Installing the toolkit.

   IBM® API Connect *<version>* Local Test Environment
   : Optional test environment. See Testing an API with the Local Test Environment

   IBM® API Connect *<version>* Security Signature Bundle File
   : Checksum files that you can use to verify the integrity of your downloads.

   Note: The Fix Pack page contains two copies of the files for each subsystem. One copy is for new installations and one is for upgrades of existing installations. For new installations, do not use files with *Upgrade* in the file name.
2. Select one of the following actions:
   - If you are configuring your first subsystem, go to step 3.
   - If you have already configured a subsystem, go to step 5.
3. Install the installation utility.
   a. Locate the apicup installation utility file for your operating system,.and download it:
   b. Rename the file for your OS type to **apicup**. Note that the instructions in this documentation refer to **apicup**.
   c. OSX and Linux® only: Make the apicup file an executable file by entering the following command:

      ```
      chmod +x apicup
      ```

   d. Set your path to the location of your apicup file.
      - For the OSX and Linux operating systems:

        ```
        export PATH=$PATH:/Users/your_path/
        ```

      - For the Windows operating system:

        ```
        set PATH=c:\your_path;%PATH%
        ```

4. Create a project called *myProject* and copy the **apicup** executable into the project directory:
   a. Create a project:

      ```
      apicup init myProject
      ```

   b. Copy the **apicup** executable into the *myProject* directory or folder created by the **apicup init** command.
   c. From within your project directory, specify the API Connect license:

      ```
      apicup licenses accept <License_ID>
      ```

      The <***License_ID***> is specific to the API Connect Program Name you purchased. To view the supported license IDs, see API Connect licenses.

   Important:
   Use a single APICUP project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.

   The original project directory created with APICUP during the initial product installation (for example, *myProject*) is required to both restore the database and to upgrade your deployment. You cannot restore the database or perform an upgrade without the initial project directory because it contains pertinent information about the cluster. Note that the endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored or upgraded system. A good practice is to backup the original project directory to a location from where it can always be retrieved.

   Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.

The apiconnect-up-v10.yml file is created in that directory.

Important: The apiconnect-up-v10.yml file must be in a secure and permanent location. The file contains password information and other information that is exposed in text format. Ensure that the project directory is secure.

5. Continue with the configuration steps for your subsystem:
   - Deploying the Management virtual appliance
   - Deploying the Analytics virtual appliance
   - Deploying the Developer Portal virtual appliance

Verifying the integrity of IBM product files

You can verify the integrity of files to ensure that they originated from IBM and are not modified.

Before you begin

Import the public key so you can use it to verify files. You only need to import the public key once.

1. Install **gpg** (GNU Privacy Guard), available at https://www.gnupg.org/.
2. Create a file called PRD0003216key.pub.gpg and copy the following content into the file:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGP+Fr0BEADz5ppiT+RhdOw+PihTcMirn4umuinsTU66qpWZES+aLmSjw3Ik
BUlD0GQkJDn6qqZyMhYUwaLVicTtLrChykzfkd3Cdzis0WRsCJNdpuLIV9OS8L9a
zCMn09lHQgaf4+eYpDYOm5OBmRP+wqP3iJrC9dVho8oRaFUBmWi30Hak+YE8xBDU
bsigBCRuLxvtQJUTHXYK8lXdRQQz2acxTeo2+ozAsR594QZSCPuv9dBV93tKvbG0
mzLyrWUZKOACZuY5Uh2SVsvnwC5jZYd+KdaffzrdgR8/MFMKAIPsQNFExta5zaiN
ENhtwZd+XsUF7Jngij/oC30GI58WEP+iRHUduhx/olmqyVnOt+yeY+qtxCwss+88
azChYSdlGP540BqUZIcM1IvibbL8LqrCcOpOXCuHG/z44co1Ic1r9bveHYUUrhtv
kXS1fLQF3y44Oz+zf9rEah8CwmL5mQxqOeGszx+2bygxajqBJH28+H/MTMgbMsDJ
Y5M1PyojQLcv5+PO7vNs0w+IBWKzyJIFoSXI3jzGR53vXlyUJ3JgAGXQiIdo7LZl
Wa6WFoJkOnESRjfY00MRIoPdtvEJ99mzr09vwEvhEPsJfo0LkOGAkjjAq6E6Ihto
BBX+YlAASupnIuJBg9uV790YuCBPkDsQ+JrgIUTShE3Vu6MGMNfL3cxpxwARAQAB
tB1BUElDb25uZWN0IDxwc2lydEB1cy5pYm0uY29tPokCOgQTAQgAJAUCY/4WvQIb
DwULCQgHAgYVCgkICwIEFgIDAQIeAQUJAAAAAAAKCRACDta12+ZfO3RlD/9fl0cH
R68313LP9we/BSMjEzv+Ik3R8/nGlZ+huvekaNQp9nJxyhU3vkF89N6NTEzh7m78
m7dpRkwnJDNyd/AP7ACvqjqYSXUx49sw3ctyWxMqtiRaQb9ChR8vLIrybmQBoukQ
9p9h9DN+sNPz55AubZ/2kIu8mXDSgALyM3ULTqFRydAXtCNQ0/+iov443nvyyfSM
drfJOfi7xvxNnRGkubRnAC3Qyyyz/k/H9cb0HRmXKQxrxLwGuRiz0MrbYu9tqCJN
Zm9+Xs/AsW5R5iGu4aPpVIMayUWlU8fWhPmapqrN0JlGsfid38WXiCtYAqX8PbbT
vzy4ZDx6aJiPsKb66DuhmS5OJjvud480uTBaJBx7PKdqlL+VN7zjPQ8TKCO/qMcc
lLrBsVavQrltLHi4EU8QzZyfIYvAMQZ9gL9FXqhCXUo1rz4qHytOikeK3XaStI+D
639LPfuiw+BJwDtLuJIaMVUyb+U/XBvCRGYmyZ8k3UUgDSj0wbtBHdXt4zSLdEhR
n1jegzLd248igap8bzDM0cWaji4YvvxLlOE2K9wWJj+cCU4+3PJuLv8wKmfIWp6j
Dv4il41RuROWkVJrj+U35B6dVGM7xOqk8WyFL548M7jimrs0gI+j2wxrKZtkqqOf
SzD8H00SJnD4HLhYmaatHuhiLJkXwB1wHeLO5g==
=1Zs0
-----END PGP PUBLIC KEY BLOCK-----
```

3. Run the following command to import the public key to **gpg**:

```
gpg --import PRD0003216key.pub.gpg
```

For example:

```
$ gpg --import PRD0003216key.pub.gpg
gpg: key 020ED6B5DBE65F3B: public key "APIConnect <psirt@us.ibm.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

4. Run the following command to verify that the key was imported successfully:

```
gpg --list-key --fingerprint APIConnect
```

Make sure that the output of the command matches the following information:

```
$ gpg --list-key --fingerprint APIConnect
pub   rsa4096 2023-02-28 [SCE]
      39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
uid           [ unknown] APIConnect <psirt@us.ibm.com>
```

Procedure

Complete the following steps to verify product files for API Connect:

1. Visit IBM Fix Central and download the appropriate signatures_*<version>*.zip file for the version of API Connect that you will install or upgrade to. Extract the signatures_*<version>*.zip file to the folder containing the files that you want to verify.
2. Verify individual API Connect files to compare each file's signature with the fingerprint of the public key.
   To verify a file, run the following command:

```
gpg --verify <file_to_verify>.asc
```

For example, to verify helper_files.zip, ensure that it is stored in the same folder as the signatures file and run the following command:

```
gpg --verify helper_files.zip.asc
```

Example of a successful verification:

```
$ gpg --verify helper_files.zip.asc
gpg: assuming signed data in 'helper_files.zip'
gpg: Signature made Mon 28 Sep 17:35:17 2020 PDT
gpg:                using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
gpg: Good signature from "APIConnect <psirt@us.ibm.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 39ED 1634 5FCD EDB5 D6ED  E860 020E D6B5 DBE6 5F3B
```

Example of a failed verification:

```
gpg: assuming signed data in 'helper_files.zip'
gpg: Signature made Tue 11 Jul 2023 09:28:02 PM PDT
gpg:                using RSA key 39ED16345FCDEDB5D6EDE860020ED6B5DBE65F3B
gpg: BAD signature from "APIConnect <psirt@us.ibm.com>" [unknown]
```

Deploying the Management virtual appliance

You can create a virtual server by deploying the relevant IBM® API Connect OVA file on a VMWare virtual server. Create all of the virtual servers that you want to use in your cloud.

Before you begin

Before you deploy:

- Review the Deployment requirements on VMware.
- Review the API Connect configuration on VMware.
- Review What's new for v10 installation
- For information on deploying a cluster, see Configuring API Connect subsystems in a cluster
- If you are upgrading from a previous version, see Upgrading API Connect.

About this task

You must deploy the API Connect OVA template to create each Management virtual server that you want in your cloud.

Attention: The following settings apply to features that are currently not supported, and should not be modified:

- Subsystem settings: test-and-monitor-enabled
- Certificate settings: hub, turnstile

Procedure

Complete each of the following tasks in order:

1. Configuring the Management subsystem
2. Deploying the Management subsystem OVA file
3. Verify installation of the Management subsystem

What to do next

If you want to deploy an API Connect Analytics OVA file, continue with Deploying the Analytics virtual appliance.

If you want to deploy an API Connect Developer Portal OVA file, continue with Deploying the Developer Portal virtual appliance.

Identify the DataPower® appliances to be used as gateway servers in the API Connect cloud and obtain the IP addresses.

- **Configuring the Management subsystem**
  Specify configuration properties for your management subsystem, and create an ISO.
- **Deploying the Management subsystem OVA file**
  After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.
- **Verify installation of the Management subsystem**
  Verify that deployment of the management subsystem OVA file succeeded.

Configuring the Management subsystem

Specify configuration properties for your management subsystem, and create an ISO.

Before you begin

Review Deploying the Management virtual appliance

About this task

Use the **apicup** installation utility to specify configuration settings for your management subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in First steps for deploying in a VMware environment.
2. Change to the project directory.

   ```
   cd myProject
   ```

3. Create a management subsystem.

   ```
   apicup subsys create mgmt management
   ```

   Where:
   - *mgmt* is the name of your management server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
   - **management** indicates that you are creating a management microservice.

   The API Connect Helm charts are deployed into the default namespace. You do not need to specify a namespace.

   Tip: At any time, you can view the current management subsystem values in the apiconnect-up-v10.yml by running the **apicup subsys get** command:

   ```
   apicup subsys get mgmt
   ```

   If you have not yet configured the subsystem, the command might return errors. Also, if you have not updated the value, a default value is listed, if there is one that is available.
   After configuration is complete, you can view output similar to the following sample:

```
Appliance settings
==================

Name                            Value
Description
----                            -----
------
additional-cloud-init-file
(Optional) Path to additional cloud-init yml file
data-device                     sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld5l8TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/
(Optional) Console login password for `apicadm` user, password must be pre-hashed
dns-servers                     [8.8.8.8]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network                 172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network             172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                    eth0
Device for API/UI traffic (Eg: eth0)
search-domain                   [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                    [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                   eth0
Device for cluster traffic (Eg: eth0)


Subsystem settings
==================

Name                            Value
Description
----                            -----
------
deployment-profile              n1xc4.m16
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for
portal
license-use                     production
License use (production/nonproduction)
multi-site-ha-enabled           false
Multi site HA enabled
multi-site-ha-mode              active
Multi site HA mode (active/passive)
replication-peer-fqdn
Replication peer fully qualified name (replication endpoint of active mode site)
site-name
Site name, used in k8s resource names
test-and-monitor-enabled        false
Test and Monitor enabled


Endpoints
=========

Name                            Value
Description
----                            -----
------
api-manager-ui                  api-manager-ui.testsrv0231.subnet1.example.com
FQDN of API manager UI endpoint
```

```
cloud-admin-ui                cloud-admin-ui.testsrv0231.subnet1.example.com
FQDN of Cloud admin endpoint
consumer-api                  consumer-api.testsrv0231.subnet1.example.com
FQDN of consumer API endpoint
hub
FQDN of Test and Monitor hub endpoint, only required if Test and Monitor is enabled
management-replication
FQDN of Management replication endpoint, only required if HA is enabled
platform-api                  platform-api.testsrv0231.subnet1.example.com
FQDN of platform API endpoint
turnstile
FQDN of Test and Monitor turnstile endpoint, only required if Test and Monitor is enabled
```

4. Specify your deployment profile, the available profiles are described here: Planning your deployment topology and profiles.

   `apicup subsys set mgmt deployment-profile=n1xc4.m16`

   Note: The deployment profiles that are shown in the **`Description`** column of the **`apicup get`** output are not correct. The available profiles are documented in Planning your deployment topology and profiles.
5. Specify the API Connect license information:
   a. Specify the license ID:

      `apicup licenses accept <License_ID>`

      The <***License_ID***> is specific to the API Connect Program Name you purchased. To view the supported license IDs, see API Connect licenses.

   b. Specify the license type:
      `apicup subsys set mgmt`
      `license-use=<license_type>`

      The *license_type* must be either **`production`** or **`nonproduction`**. If not specified, the default value is **`nonproduction`**.

6. Configure scheduled backups for your management database.
   Use **`apicup subsys set`** to set the following backup properties:

Table 1. Configuration settings for backup of management subsystem database

Parameter	Description
`database-backup-auth-pass`	The password for the server specified in **`database-backup-host`**. This is the S3 Secret Access Key parameter. The password is converted and stored in Base64 encoded format. For example: **`apicup subsys set <subsystem_name> database-backup-auth-pass '<password>'`** Note that you cannot use the `=` sign to assign the password to **`database-backup-auth-pass`**. If you change this value later, to apply the change you must run **`apicup subsys db-reload <subsystem_name>`** after **`apicup subsys install <subsystem_name>`**
`database-backup-auth-user`	The username for the server specified in **`database-backup-host`**. This is the S3 Secret Key ID. **`apicup subsys set <subsystem_name> database-backup-auth-user <username>`** If you change this value later, to apply the change you must run **`apicup subsys db-reload <subsystem_name>`** after **`apicup subsys install <subsystem_name>`**
`database-backup-host`	The backups host. Specify S3 endpoint with the corresponding S3 region in the format **`<S3endpoint>/<S3region>`**: • **`apicup subsys set <subsystem_name> database-backup-host=s3.ibm.com/eu-west-2`** • **`apicup subsys set <subsystem_name> database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard`**
`database-backup-path`	The path to the location of the backup. Specify the name of your S3 bucket to store backup data, in the format **`bucket-name/folder`**. The use of subdirectories in the bucket name is not supported. When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same S3 bucket name in the database backup configurations. Each management subsystem must use a unique S3 bucket name. Ensure that **`bucket-name/folder`** is empty. **`apicup subsys set <subsystem_name> database-backup-path=bucket-name/apic-backup`**

Parameter	Description
`database-backup-schedule`	Cron schedule for performing automatic backups. The format for the schedule is: • ****** • ------ • |||||| • ||||| +----- day of week (0 - 6) (Sunday=0) • |||| +------- month (1 - 12) • ||| +--------- day of month (1 - 31) • || +----------- hour (0 - 23) • | +------------- min (0 - 59) • +--------------- sec (0 - 59) The default is an empty string, which means no backups are taken. The timezone for backups is UTC. This example will schedule a back up daily at 1am. **`apicup subsys set <subsystem_name> database-backup-schedule="0 0 1 * * *"`**
`backup-certs`	**Optional**. Backup certificates to use for TLS communication between API Connect and the backup server. For example: **`apicup subsys set <subsystem_name> backup-certs=<certificate_name>`**
`database-backup-s3-uri-style`	**Optional**: Specifies whether the URI is in the form of `host` or `path`. **`apicup subsys set <subsystem_name> database-backup-s3-uri-style='<host|path>'`** Supported values: • **`host`** • **`path`**
`database-backup-repo-retention-full`	The number of days of backups to retain. For example, if **`database-backup-repo-retention-full`** is set to 30, then backups for the last 30 days are retained. Older backups are deleted. After a restore, the count of the number of days is reset to zero. Minimum value: 1 Maximum value: 9999999 Default: none

Note: If you do not have an S3 object-store available, you can install the management subsystem with no backup settings configured, by setting **`accept-no-backups`** to **`true`**:

```
apicup subsys set mgmt accept-no-backups=true
```

If you use this configuration, then no management database backups are taken, and disaster recovery is impossible.

7. Optional: Configure your logging.
   Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.
   a. Complete the procedure at Configuring remote logging for a VMware deployment.
   b. Enter the following command to create the log file:

   ```
   apicup subsys set mgmt additional-cloud-init-file=config_file.yml
   ```

8. Enter the following commands to update the apiconnect-up-v10.yml with the information for your environment:
   a. Set your search domain. Multiple search domains should be separated by commas.

   ```
   apicup subsys set mgmt search-domain=your_search_domain
   ```

   Where *your_search_domain* is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is *mycompany.example.com*.

   Ensure that the value for *your_search_domain* is resolved in the system's /etc/resolv.conf file to avoid "502" errors when accessing the Cloud Manager web site. For example:

   ```
   # Generated by resolvconf
   search your_search_domain ibm.com other.domain.com
   ```

   b. Set your domain name servers (DNS).
   Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

   ```
   apicup subsys set mgmt dns-servers=ip_address_of_dns_server[,ip_address_of_another_dns_server_if_necessary]
   ```

   c. Use `apicup` to set your endpoints.
   You can use wildcard aliases or host aliases with your endpoints.
   Optionally, you can specify all endpoints with one **`apicup`** command.

   Note: You cannot specify the underscore character "_" in domain names that are used in endpoints. See API Connect configuration on VMware.

Table 2. Management subsystem endpoints

Setting	Endpoint host description
`platform-api`	Platform API endpoint. The host where your platform API calls are routed. **`apicup subsys set mgmt platform-api=platform-api.hostname.domain`**
`consumer-api`	Consumer API endpoint. The host where your consumer API calls are routed. **`apicup subsys set mgmt consumer-api=consumer-api.hostname.domain`**
`cloud-admin-ui`	Cloud admin user interface API endpoint. The host where your cloud administrator user-interface API calls are routed. **`apicup subsys set mgmt cloud-admin-ui=cloud-admin-ui.hostname.domain`**

Setting	Endpoint host description
`api-manager-ui`	API Manager user interface endpoint. The host where your API Manager API calls are routed. `apicup subsys set mgmt api-manager-ui=api-manager-ui.hostname.domain`
hub	The core microservice components for the Automated API behavior testing capability. This is an optional component and can be left unset if not used. `apicup subsys set mgmt hub hub.hostname.domain` Note: Automated API behavior testing is not supported in two data center disaster recovery configurations.
turnstile	Allows for communication between the Test pods and the API Management subsystem. This is part of the Automated API behavior testing capability. This is an optional component and can be left unset if not used. `apicup subsys set mgmt turnstile turnstile.hostname.domain` Note: Automated API behavior testing is not supported in two data center disaster recovery configurations.

9. Set a Public key.

   `apicup subsys set mgmt ssh-keyfiles=path_to_public_ssh_keyfile`

   Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check in later in Deploying the Management subsystem OVA file.

10. Set the password that you enter to log into your Management appliance for the first time.
    a. **Important**: Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.
    b. If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the **mkpasswd** command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: `gem install mkpasswd` .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

    c. Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	`mkpasswd --method=sha-512 --rounds=4096 password`
Windows, Red Hat	For example, using OpenSSL: `openssl passwd -1 password`. Note that you might need to add your password hashing utility to your path; for example, on Windows: `set PATH=c:\cygwin64\bin;%PATH%`

    d. Set the hashed password for your subsystem:

       `apicup subsys set mgmt default-password='hashed_password'`

11. Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.
    You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See API Connect configuration on VMware.
    a. Update the IP range for the Kubernetes pod

       `apicup subsys set mgmt k8s-pod-network='new_pod_range'`

       Where *new_pod_range* is the new value for the range.

    b. Update the IP range for Service networks.

       `apicup subsys set mgmt k8s-service-network='new_service_range'`

       Where *new_service _range* is the new value for the range.

12. Add your hosts.

    `apicup hosts create mgmt hostname.domainname hd_password`

    Where the following are true:
    - *hostname.domainname* is the fully qualified name of the server where you are hosting your Management service, including the domain information.
    - *hd_password* is the password that the Linux Unified Key Setup uses to encrypt the storage for your Management service. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in `apiconnect-up-v10.yml`.

    Repeat this command for each host that you want to add.

    Note: Host names cannot be changed on a cluster after the initial installation.

13. Create your interfaces.

    `apicup iface create mgmt hostname.domainname physical_network_id host_ip_address/subnet_mask network_gateway_ip_address`

    Where *physical_network_id* is the network interface ID of your physical server. The value is most often `eth0`. The value can also be `ethx`, where `x` is a number identifier.
    The format is similar to this example: `apicup iface create mgmt myHostname.domain eth0 192.0.2.10/255.255.255.0 192.0.2.1`.

Note: The ***network_gateway_ip_address*** is the network gateway (not a DataPower Gateway). If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway.

14. Optional: Use **apicup** to view the configured hosts:

```
apicup hosts list mgmt
testsrv0231.subnet1.example.com
    Device  IP/Mask                    Gateway
    eth0    1.2.152.231/255.255.254.0  1.2.152.1
```

Note: This command might return the following messages, which you can ignore:

```
* host is missing traffic interface
* host is missing public interface
```

Note: If you are configuring two data center deployment, continue with the installation instructions in Installing a two data center deployment.

15. Optional: Verify that the configuration settings are valid.

```
apicup subsys get mgmt --validate
```

The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

```
Appliance settings
==================

Name                          Value
----                          -----
additional-cloud-init-file
✔
data-device                   sdb
✔
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld5l8TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/   ✔
dns-servers                   [8.8.8.8]
✔
extra-values-file
✔
k8s-pod-network               172.16.0.0/16
✔
k8s-service-network           172.17.0.0/16
✔
public-iface                  eth0
✔
search-domain                 [subnet1.example.com]
✔
ssh-keyfiles                  [/home/vsphere/.ssh/id_rsa.pub]
✔
traffic-iface                 eth0
✔


Subsystem settings
==================

Name                          Value
----                          -----
deployment-profile            n1xc4.m16
✔
license-use                   production
✔
multi-site-ha-enabled         false
✔
multi-site-ha-mode            active
✔
replication-peer-fqdn
✔
site-name
✔
test-and-monitor-enabled      false
✔


Endpoints
=========

Name                          Value
----                          -----
api-manager-ui                api-manager-ui.testsrv0231.subnet1.example.com
✔
cloud-admin-ui                cloud-admin-ui.testsrv0231.subnet1.example.com
✔
consumer-api                  consumer-api.testsrv0231.subnet1.example.com
✔
hub
✔
management-replication
✔
platform-api                  platform-api.testsrv0231.subnet1.example.com
✔
turnstile
✔
```

16. Create your ISO file.

    ```
    apicup subsys install mgmt --out mgmtplan-out
    ```

    The **--out** parameter and value are required.

    In this example, the ISO file is created in the myProject/*mgmt*plan-out directory.
    If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	`export PATH=$PATH:/Users/your_path/`
Windows, Red Hat	`set PATH="c:\Program Files (x86)\cdrtools";%PATH%`

17. Deploy the subsystem ISO. Continue with Deploying the Management subsystem OVA file.

Deploying the Management subsystem OVA file

After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.

Before you begin

You must have completed the configuration of the project file entries for the management subsystem, in Configuring the Management subsystem.

About this task

Procedure

1. Log into the VMWare vSphere Web Client.
2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
3. Right-click the directory and select Deploy OVF Template.
4. Complete the Deploy OVF Template wizard.
    a. Select the Management subsystem .OVA file template by navigating to the location where you downloaded the file in First steps for deploying in a VMware environment.
    b. Enter a name and location for your file.
    c. Select a resource for your template.
    d. Review the details for your template.
    e. Select the size of your configuration.
    f. Select the storage settings.
    Important: Select at least 200 GB for the data disk size. Note that the boot disk needs 100 GB, so you need a total of 300 GB for the API Connect management subsystem.
    g. Select the networks.
    h. Customize the Template, if necessary.
    i. Review the details to ensure that they are correct.
    j. Select Finish to deploy the virtual machine.

    Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See Requirements for initial deployment on VMware.
    The template creation appears in your Recent Tasks list.
5. Select the Storage tab in the Navigator.
6. Navigate to your datastore.
7. Upload your ISO file.
    a. Select the Navigate to the datastore file browser icon in the icon menu.
    b. Select the Upload a file to the Datastore icon in the icon menu.
    c. Navigate to the ISO file that you created in your project.
    It is the myProject/*mgmt*plan-out
    d. Upload the ISO file to the datastore.
8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
9. Locate and select your virtual machine.
10. Select the Configure tab in the main window.
11. Select Edit....
    a. On the Virtual Hardware tab, select CD/DVD Drive 1.
    b. For the Client Device, select Datastore ISO File.
    c. Find and select your datastore in the Datastores category.
    d. Find and select your ISO file in the Contents category.
    e. Select OK to commit your selection and exit the Select File window.
    f. Ensure that the Connect At Power On check box is selected.
    Tip:
        - Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
        - Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On
    g. Select OK to commit your selection and close the window.
12. Start the virtual machine by selecting the play button on the icon bar.
    The installation might take several minutes to complete, depending on the availability of the system and the download speed.
13. Verify the API Connect installation. Continue with Verify installation of the Management subsystem

Verify installation of the Management subsystem

Verify that deployment of the management subsystem OVA file succeeded.

Before you begin

You must have completed Deploying the Management subsystem OVA file.

About this task

After you configure an ISO and deploy it with the management subsystem ova, as described in Deploying the Management subsystem OVA file, verify that the subsystem installed correctly and is functional.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
   a. Enter the following command to connect to *mgmt* using SSH:

      ```
      ssh ip_address -l apicadm
      ```

      You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.
   b. Select Yes to continue connecting.
      Your host names are automatically added to your list of hosts.
   c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
      Note that after installation completes, it can take several minutes for all servers to start. If you see the error message **`Subsystems not running`**, wait a few minutes, try the command again, and review the output in the **`Status`** column.

      The command output for a correctly running Management system is similar to the following lines:

      ```
      apicadm@testsys0181:~$ sudo apic status

      INFO[0001] Log level: info
      Cluster members:
      - testsys0164.subnet1.example.com (1.1.1.1)
        Type: BOOTSTRAP_MASTER
        Install stage: DONE
        Upgrade stage: NONE
        Docker status:
          Systemd unit: running
        Kubernetes status:
          Systemd unit: running
          Kubelet version: testsys0164 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
        Etcd status: pod etcd-testsys0164 in namespace kube-system has status Running
        Addons: calico, dns, helm, kube-proxy, metrics-server, nginx-ingress,
      - testsys0165.subnet1.example.com (1.1.1.2)
        Type: MASTER
        Install stage: DONE
        Upgrade stage: NONE
        Docker status:
          Systemd unit: running
        Kubernetes status:
          Systemd unit: running
          Kubelet version: testsys0165 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
        Etcd status: pod etcd-testsys0165 in namespace kube-system has status Running
        Addons: calico, kube-proxy, nginx-ingress,
      - testsys0181.subnet1.exmample.com (1.1.1.3)
        Type: MASTER
        Install stage: DONE
        Upgrade stage: NONE
        Docker status:
          Systemd unit: running
        Kubernetes status:
          Systemd unit: running
          Kubelet version: testsys0181 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
        Etcd status: pod etcd-testsys0181 in namespace kube-system has status Running
        Addons: calico, kube-proxy, nginx-ingress,
      Etcd cluster state:
      - etcd member name: testsys0164.subnet1.example.com, member id: 11019072309842691371,
            cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17
      - etcd member name: testsys0165.subnet1.example.com, member id: 541472388445093633,
            cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17
      - etcd member name: testsys0181.subnet1.example.com, member id: 3261849123413063575,
            cluster id: 5154498743703662183, leader id: 11019072309842691371, revision: 21848, version: 3.1.17

      Pods Summary:

      NODE                NAMESPACE           NAME                                                    READY
      STATUS          REASON
      testsys0165         kube-system         calico-node-jp8zv                                       2/2
      Running
      testsys0164         kube-system         calico-node-pjjgh                                       2/2
      Running
      testsys0181         kube-system         calico-node-ssb9w                                       2/2
      Running
      testsys0164         kube-system         coredns-87cb95869-9nvdr                                 1/1
      Running
      ```

```
testsys0164       kube-system       coredns-87cb95869-r9q8w                                  1/1
Running
testsys0164       kube-system       etcd-testsys0164                                         1/1
Running
testsys0165       kube-system       etcd-testsys0165                                         1/1
Running
testsys0181       kube-system       etcd-testsys0181                                         1/1
Running
testsys0165       kube-system       ingress-nginx-ingress-controller-92mkz                   1/1
Running
testsys0181       kube-system       ingress-nginx-ingress-controller-kt9sr                   1/1
Running
testsys0164       kube-system       ingress-nginx-ingress-controller-p7x55                   1/1
Running
testsys0164       kube-system       ingress-nginx-ingress-default-backend-6f58fb5f56-t27gx   1/1
Running
testsys0164       kube-system       kube-apiserver-testsys0164                               1/1
Running
testsys0165       kube-system       kube-apiserver-testsys0165                               1/1
Running
testsys0181       kube-system       kube-apiserver-testsys0181                               1/1
Running
testsys0164       kube-system       kube-apiserver-proxy-testsys0164                         1/1
Running
testsys0165       kube-system       kube-apiserver-proxy-testsys0165                         1/1
Running
testsys0181       kube-system       kube-apiserver-proxy-testsys0181                         1/1
Running
testsys0164       kube-system       kube-controller-manager-testsys0164                      1/1
Running
testsys0165       kube-system       kube-controller-manager-testsys0165                      1/1
Running
testsys0181       kube-system       kube-controller-manager-testsys0181                      1/1
Running
testsys0165       kube-system       kube-proxy-7gqpw                                         1/1
Running
testsys0181       kube-system       kube-proxy-8hc8t                                         1/1
Running
testsys0164       kube-system       kube-proxy-bhgcq                                         1/1
Running
testsys0164       kube-system       kube-scheduler-testsys0164                               1/1
Running
testsys0165       kube-system       kube-scheduler-testsys0165                               1/1
Running
testsys0181       kube-system       kube-scheduler-testsys0181                               1/1
Running
testsys0164       kube-system       metrics-server-6fbfb84cdd-lffxc                          1/1
Running
testsys0164       kube-system       tiller-deploy-84f4c8bb78-xxfds                           1/1
Running
```

2. Verify you can access the API Connect Cloud Manager UI. Enter the URL in your browser.
   The syntax is **`https://<hostname.domain>/admin`**. For example:

   ```
   https://cloud-admin-ui.testsrv0231.subnet1.example.com/admin
   ```

   The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See Accessing the Cloud Manager user interface.

3. Take a backup of your project directory and configure scheduled management database backups. See Backing up and restoring the Management subsystem.
   Note: If you installed v10.0.6 and already configured database backups during installation, then verify that the configuration succeeded: Verify configuration for s3 backup V10.0.6.0.

Deploying the Analytics virtual appliance

Define your analytics subsystem, create an ISO file for each analytics node, and deploy the analytics nodes using the analytics OVA file on a VMware virtual server.

Before you begin

Before you deploy:

- Review the Deployment requirements on VMware.
- Review the API Connect configuration on VMware.
- Review What's new for v10 installation
- For information on deploying a cluster, see Configuring API Connect subsystems in a cluster
- If you are upgrading from a previous version, see Upgrading API Connect.

About this task

You must deploy the API Connect OVA template to create each analytics virtual server that you want in your cloud.

- **Configuring the Analytics subsystem**
  Specify configuration properties for your analytics subsystem, and create the ISO files.
- **Deploying the Analytics subsystem OVA file**
  After you have specified configuration settings and created your ISO files, deploy the subsystem OVA into an appliance VM.

- **Verifying deployment of the Analytics subsystem**
  Verify that deployment of the analytics subsystem and configure scheduled backups.

Configuring the Analytics subsystem

Specify configuration properties for your analytics subsystem, and create the ISO files.

Before you begin

Review Planning your analytics deployment and decide on the following:

- Deployment profile for your analytics subsystem.
- Storage space requirement for your analytics data.
- If you want to offload analytics data to a third-party system and disable local storage, you must configure storage disablement before you create the deployment ISO files. See: Disable local storage.

About this task

Use the **apicup** installation utility to specify configuration settings for your analytics subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in First steps for deploying in a VMware environment.
2. Change to the project directory.

   ```
   cd myProject
   ```

3. Create an analytics subsystem.

   ```
   apicup subsys create analyt analytics
   ```

   Where:
   - *analyt* is the name of the analytics server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
   - **analytics** indicates that you want it to create an analytics subsystem.

   The apiconnect-up-v10.yml file that is in that directory is updated to add the analytics-related entries.
   Tip: At any time, you can view the current analytics subsystem values in the apiconnect-up-v10.yml by running the **apicup get** command:

   ```
   apicup subsys get analyt
   ```

   If you have not updated the value, a default value is listed, if there is one that is available.
   Sample output from **apicup subsys get**:

```
 Appliance settings
==================

Name                                    Value           Description
----                                    -----           ------
additional-cloud-init-file                              (Optional) Path to additional cloud-init yml file
data-device                             sdb             VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password                                        (Optional) Console login password for `apicadm` user, password
must be pre-hashed
dns-servers                             []              List of DNS servers
extra-values-file                                       (Optional) Path to additional configuration yml file
k8s-pod-network                         172.16.0.0/16   (Optional) CIDR for pods within the appliance
k8s-service-network                     172.17.0.0/16   (Optional) CIDR for services within the appliance
public-iface                            eth0            Device for API/UI traffic (Eg: eth0)
search-domain                           []              List for DNS search domains
ssh-keyfiles                            []              List of SSH public keys files
traffic-iface                           eth0            Device for cluster traffic (Eg: eth0)


Subsystem settings
==================

Name                                    Value           Description
----                                    -----           ------
analytics-backup-auth-pass                              (Optional) Server password for analytics backups
analytics-backup-auth-user                              (Optional) Server username for analytics backups
analytics-backup-certs                                  (Optional) Backup certs which are used for TLS communication
between APIC and backup server. Currently only supported for s3 backups.
analytics-backup-chunk-size             1GB             analytics-backup-chunk-size
analytics-backup-host                                   (Optional) FQDN for analytics backups server
analytics-backup-path                                   (Optional) path for analytics backups server
analytics-backup-schedule               0 0 * * *       (Optional) Cron schedule for analytics backups
analytics-enable-compression            true            (Optional) Determines whether metadata files are stored in
compressed format
analytics-enable-server-side-encryption    false        (Optional) Determines whether files are encrypted. When set to
true, files are encrypted on the server side using AES256
deployment-profile                      n1xc2.m16       Deployment profile (n1xc2.m16/n3xc4.m16) for analytics,
(n1xc2.m16/n1xc4.m16/n3xc2.m16/n3xc4.m16) for management, (n1xc2.m8/n1xc4.m16/n1xc8.m16/n3xc4.m8/n3xc8.m16) for portal
license-use                             nonproduction   License use (production/nonproduction)
```

```
Endpoints
=========

Name                                        Value           Description
----                                        -----           ------
analytics-ingestion                                         FQDN of Analytics ingestion endpoint
```

4. Specify your **deployment-profile**.

   apicup subsys set *analyt* deployment-profile=*<profile name>*

   For example, to deploy with a three replica profile, run the following command:

   apicup subsys set *analyt* deployment-profile=n3xc4.m16

   Note: The deployment profiles that are shown in the **Description** column of the **apicup get** output are not correct. The available profiles are documented in Planning your deployment topology and profiles.
5. Specify your license type.

   apicup subsys set *analyt* license-use=*license_type*

   The ***license_type*** must be either **production** or **nonproduction**. If not specified, the default value is **nonproduction**.

6. Optional: Configure your logging.
   Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.
   a. Complete the procedure at Configuring remote logging for a VMware deployment.
   b. Enter the following command to create the log file:

      apicup subsys set analyt additional-cloud-init-file=config_file.yml

7. Set your ingestion endpoint.

   apicup subsys set analyt analytics-ingestion=*unique_hostname.domain*

   The ingestion endpoint must be a unique host name that points to the IP address of the OVA (single node deployment), or to the IP of a load balancer configured in front of the OVA nodes. You will use the ingestion endpoint when registering the analytics service in the Cloud Manager UI.
8. Set your search domain. Multiple search domains should be separated by commas.

   apicup subsys set analyt search-domain=*your_search_domain*

   Where *your_search_domain* is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is *mycompany.example.com*.

   Ensure that the value for *your_search_domain* is resolved in the system's /etc/resolv.conf file to avoid "502" errors when accessing the Cloud Manager web site. For example:

   ```
   # Generated by resolvconf
   search your_search_domain ibm.com other.domain.com
   ```

9. Set your domain name servers (DNS).
   Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

   apicup subsys set analyt dns-servers=*ip_address_of_dns_server*

   DNS entries may not be changed on a cluster after the initial installation.
10. Set a Public key.

    apicup subsys set analyt ssh-keyfiles=*path_to_public_ssh_keyfile*

    Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check later in Verifying deployment of the Analytics subsystem.

11. You can set the password that you enter to log into your analytics server for the first time.
    a. Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.
    b. If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the **mkpasswd** command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: **gem install mkpasswd** .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

    c. Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	**mkpasswd --method=sha-512 --rounds=4096 *password***
Windows, Red Hat	For example, using OpenSSL: **openssl passwd -1 *password***. Note that you might need to add your password hashing utility to your path; for example, on Windows: **set PATH=c:*cygwin64*\bin;%PATH%**

    d. Set the hashed password for your subsystem:

       apicup subsys set analyt default-password='*hashed_password*'

       Notes:

- The password is hashed. If it is in plain text, you cannot log into the VMWare console.
- Note that the password can only be used to login through the VMware console. You cannot use it to SSH into the Appliance as an alternative to using the **ssh-keyfiles**.
- On Linux or OSX, use single quotes around *hashed_password*. For Windows, use double quotes.
- If you are using a non-English keyboard, understand the limitations with using the remote VMware console. See Requirements for initial deployment on VMware.

12. Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.
You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See API Connect configuration on VMware.
    a. Update the IP range for the Kubernetes pod

    ```
    apicup subsys set analyt k8s-pod-network='new_pod_range'
    ```

    Where *new_pod_range* is the new value for the range.

    b. Update the IP range for Service networks.

    ```
    apicup subsys set analyt k8s-service-network='new_service_range'
    ```

    Where *new_service _range* is the new value for the range.

13. Define the hostname for each subsystem node you are deploying:

    ```
    apicup hosts create analyt hostname.domainname hd_password
    ```

    Where the following are true:
    - *hostname.domainname* is the fully qualified name of the server where you are hosting your analytics service, including the domain information.
    - *hd_password* is the password of the Linux Unified Key Setup uses to encrypt the storage for your analytics service. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in **apiconnect-up-v10.yml**.

    Repeat this command for each subsystem VM in your deployment. For example, if you are deploying a single node one replica profile then do this once, for a three replica profile you will define three VMs.

    Note: Host names and DNS entries may not be changed on a cluster after the initial installation.
14. Define the network interface for each subsystem node you are deploying:

    ```
    apicup iface create analyt hostname.domainname physical_network_id host_ip_address/subnet_mask gateway_ip_address
    ```

    Where *physical_network_id* is the network interface ID of your physical server. The value is most often **eth0**. The value can also be **ethx**, where ***x*** is a number identifier.
    The format is similar to this example: **apicup iface create analyt myHostname.domain eth0 192.0.2.1/255.255.1.1 192.0.2.1**
    Repeat this command for each subsystem VM in your deployment. For example, if you are deploying a single node one replica profile then do this once, for a three replica profile you will define three VMs.

15. Optional: Use **apicup** to view the configured hosts:

    ```
    apicup hosts list analyt
    testsrv0233.subnet1.example.com
        Device  IP/Mask                    Gateway
        eth0    1.2.152.233/255.255.254.0  1.2.152.1
    ```

16. Optional: Enable JWT security instead of mTLS for communication from management and gateway to your analytics subsystem.
JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see Enable JWT instead of mTLS.
To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

    ```
    apicup subsys get <management subsystem>

    ...
    jwks-url     https://appliance1.apic.acme.com/api/cloud/oauth2/certs  JWKS URL for Portal and analytics subsystems to
    validate JWT -- this is unsettable and is generated based on the platform-api endpoint
    ...
    ```

    Disable mTLS and enable JWT by setting the jwks-url with **apicup**:

    ```
    apicup subsys set analyt mtls-validate-client=false
    apicup subsys set analyt jwks-url=https://appliance1.apic.acme.com/api/cloud/oauth2/certs
    ```

    Note: Do not disable mTLS without enabling JWT.
    JWT for gateway to analytics communication requires an additional step during gateway registration. Enable the Use JWT switch when you register the gateway in the Cloud Manager UI.

17. Verify that the configuration settings are valid.

    ```
    apicup subsys get analyt --validate
    ```

    The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

    ```
     apicup subsys get analyt --validate
    Appliance settings
    ==================

    Name                          Value
    ----                          -----
    additional-cloud-init-file                                                                  ✔
    data-device                   sdb                                                           ✔
    default-password              $6$rounds=4096$iMCJ9cfhFJ8X$pbmAl9ClWzcYzH
    ```

```
                                    ZFoQ6n7OnYCf/owQZIiCpAtWazs/FUn/uE8uLD.9jwHE0AX4upFSqx/jf0ZmDbHPZ9bUlCY1  ✔
dns-servers                         [1.2.3.1]                                                                 ✔
extra-values-file                                                                                             ✔
k8s-pod-network                     172.16.0.0/16                                                             ✔
k8s-service-network                 172.17.0.0/16                                                             ✔
public-iface                        eth0                                                                      ✔
search-domain                       [subnet1.example.com]                                                     ✔
ssh-keyfiles                        [/home/vsphere/.ssh/id_rsa.pub]                                           ✔
traffic-iface                       eth0                                                                      ✔

Subsystem settings
==================

Name                                      Value
----                                      -----
analytics-backup-auth-pass                                                                                    ✔
analytics-backup-auth-user                                                                                    ✔
analytics-backup-certs                                                                                        ✔
analytics-backup-chunk-size               1GB                                                                 ✔
analytics-backup-host                                                                                         ✔
analytics-backup-path                                                                                         ✔
analytics-backup-schedule                 0 0 * * *                                                           ✔
analytics-enable-compression              true                                                                ✔
analytics-enable-server-side-encryption   false                                                               ✔
deployment-profile                        n1xc2.m16                                                           ✔
license-use                               production                                                          ✔


Endpoints
=========

Name                          Value
----                          -----
analytics-ingestion           a7s-in.testsrv0233.subnet1.example.com                                          ✔
```

18. If you configured additional deployment options in an extra-values file, run the following command to make the file available during installation:

    ```
    apicup subsys set analyt extra-values-file path/analytics-extra-values.yaml
    ```

    where *analytics-extra-values*.yaml is the name of the file containing your deployment settings.

19. Create your ISO files:

    ```
    apicup subsys install analyt --out analytplan-out
    ```

    The **--out** parameter and value are required. In this example, the ISO files are created in the myProject/analytplan-out directory.

    There will be one ISO file for each VM host you defined in step 13.
    If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	`export PATH=$PATH:/Users/your_path/`
Windows, Red Hat	`set PATH="c:\Program Files (x86)\cdrtools";%PATH%`

20. Continue with Deploying the Analytics subsystem OVA file.

Deploying the Analytics subsystem OVA file

After you have specified configuration settings and created your ISO files, deploy the subsystem OVA into an appliance VM.

About this task

Repeat this procedure for each analytics VM you are deploying. Use the ISO files created in Configuring the Analytics subsystem.

Procedure

1. Log into the VMWare vSphere Web Client.
2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
3. Right-click the directory and select Deploy OVF Template.
4. Complete the Deploy OVF Template wizard.
   a. Select the analytics subsystem .OVA file template by navigating to the location where you downloaded the file in First steps for deploying in a VMware environment.
   b. Enter a name and location for your file.
   c. Select a resource for your template.
   d. Review the details for your template.
   e. Select the size of your configuration.
   f. Select the storage settings.
      Note: Ensure you specify a storage size large enough for your estimated analytics data usage: Estimating storage requirements. If in doubt, set size to 500GB.
   g. Select the networks.
   h. Customize the Template, if necessary.
   i. Review the details to ensure that they are correct.

j. Select Finish to deploy the virtual machine.

Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See Requirements for initial deployment on VMware.

The template creation appears in your Recent Tasks list.

5. Select the Storage tab in the Navigator.
6. Navigate to your datastore.
7. Upload your ISO file.
   a. Select the Navigate to the datastore file browser icon in the icon menu.
   b. Select the Upload a file to the Datastore icon in the icon menu.
   c. Navigate to the ISO file that you created in your project.
      It is the myProject/*analyt*plan-out
   d. Upload the ISO file to the datastore.
8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
9. Locate and select your virtual machine.
10. Select the Configure tab in the main window.
11. Select Edit....
    a. On the Virtual Hardware tab, select CD/DVD Drive 1.
    b. For the Client Device, select Datastore ISO File.
    c. Find and select your datastore in the Datastores category.
    d. Find and select your ISO file in the Contents category.
    e. Select OK to commit your selection and exit the Select File window.
    f. Ensure that the Connect At Power On check box is selected.
       Tip:
       - Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
       - Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On
    g. Select OK to commit your selection and close the window.
12. Start the virtual machine by selecting the play button on the icon bar.
    The installation might take several minutes to complete, depending on the availability of the system and the download speed.
13. Next, verify the deployment. Continue with Verifying deployment of the Analytics subsystem.

Verifying deployment of the Analytics subsystem

Verify that deployment of the analytics subsystem and configure scheduled backups.

Before you begin

Complete Deploying the Analytics subsystem OVA file.

About this task

After you configure an ISO and deploy it with the analytics subsystem OVA, as described in Deploying the Analytics subsystem OVA file, verify that the subsystem installed correctly and is functional, and configure the scheduled backups.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
   a. Enter the following command to connect to *analyt* using SSH:

      ```
      ssh ip_address -l apicadm
      ```

      You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.
   b. Select Yes to continue connecting.
      Your host names are automatically added to your list of hosts.
   c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
      The command output for a correctly running analytics system is similar to the following:

      ```
      #sudo apic status
      INFO[0000] Log level: info

      Cluster members:
      - apimdev0239.subnet1.example.com (9.20.152.239)
        Version: 4.4.1
        Type: BOOTSTRAP_MASTER
        Install stage: DONE
        Upgrade stage: UPGRADE_DONE
        Subsystem type: analytics
        Subsystem version: 10.0.5.1
        Subsystem detail:
        Reboot Required: false
        Containerd status:
          Systemd unit: running
        Kubernetes status:
          Systemd unit: running
          Kubelet version: apimdev0239 (5.4.0-122-generic) [Kubelet v1.23.5, Proxy v1.23.5]
        Etcd status: pod etcd-apimdev0239 in namespace kube-system has status Running
        Addons: calico, cert-manager-v1, dns, kube-proxy, local-volume-provisioner, metrics-server,
        Disk Space Available: 226G/245G | Used: 4%
        Inodes Available: 16M/16M | Used: 1%
        Containers: 21
          - Running: 21
      ```

```
    - Exited: 7
    - Unknown: 0
  Images: 32
Etcd cluster state:
- etcd member name: apimdev0239.subnet1.example.com, member id: 15210952383217851221, cluster id:
8720375562272123817, leader id: 15210952383217851221, revision: 2538453, version: 3.5.4

Pods Summary:
NODE              NAMESPACE         NAME                                          READY       STATUS
REASON
apimdev0239       certmanager       cert-manager-84f999d4c7-jz8p6                 1/1         Running
apimdev0239       certmanager       cert-manager-cainjector-5dc89f744d-7mbms      1/1         Running
apimdev0239       certmanager       cert-manager-webhook-7497758675-vn4hp         1/1         Running
apimdev0239       default           cpd-analytics-director-7d584d8ffd-qts9c       1/1         Running
apimdev0239       default           cpd-analytics-ingestion-0                     1/1         Running
apimdev0239       default           cpd-analytics-mtls-gw-f647687d6-9rbpd         1/1         Running
apimdev0239       default           cpd-analytics-osinit-md7cn                    0/1         Succeeded
apimdev0239       default           cpd-analytics-storage-0                       1/1         Running
apimdev0239       default           ibm-apiconnect-8cd7cb6bd-vjwv2                1/1         Running
apimdev0239       kube-system       calico-kube-controllers-789ff9469b-xrttr      1/1         Running
apimdev0239       kube-system       calico-node-8tfrx                             1/1         Running
apimdev0239       kube-system       coredns-79f7dbf799-7dmn6                      1/1         Running
apimdev0239       kube-system       coredns-79f7dbf799-9qgnk                      1/1         Running
apimdev0239       kube-system       etcd-apimdev0239                              1/1         Running
apimdev0239       kube-system       ingress-nginx-ingress-controller-96k8k        1/1         Running
apimdev0239       kube-system       kube-apiserver-apimdev0239                    1/1         Running
apimdev0239       kube-system       kube-apiserver-proxy-apimdev0239              1/1         Running
apimdev0239       kube-system       kube-controller-manager-apimdev0239           1/1         Running
apimdev0239       kube-system       kube-proxy-j4lbz                              1/1         Running
apimdev0239       kube-system       kube-scheduler-apimdev0239                    1/1         Running
apimdev0239       kube-system       local-volume-provisioner-rlfjs                1/1         Running
apimdev0239       kube-system       metrics-server-d9fffcbc9-nmcgr                1/1         Running
```

2. If you have now installed all subsystems, continue to Post-deployment steps.
3. Configure backups of the subsystem. Follow the instructions in Backing up and restoring the Analytics database on VMware.
   Important: It is highly recommend that you configure backups and also take additional steps to ensure that your configuration and data can be restored in the event of a disaster event. See Preparing the Analytics subsystem for disaster recovery on VMware.

Deploying the Developer Portal virtual appliance

You create a Developer Portal node by deploying the Developer Portal OVA template. After you deploy the Developer Portal OVA template, you can install the Developer Portal.

Before you begin

Before you deploy:

- Review the Deployment requirements on VMware.
- Review the API Connect configuration on VMware.
- Review What's new for v10 installation
- For information on deploying a cluster, see Configuring API Connect subsystems in a cluster
- If you are upgrading from a previous version, see Upgrading API Connect.

Note:
Ensure that your kernel or Kubernetes node has the value of its **`inotify`** watches set high enough so that the Developer Portal can monitor and maintain the files for each Developer Portal site. If set too low, the Developer Portal containers might fail to start or go into a **`non-ready`** state when this limit is reached. If you have many Developer Portal sites, or if your sites contain a lot of content, for example, many custom modules and themes, then a larger number of **`inotify`** watches are required. You can start with a value of 65,000, but for large deployments, this value might need to go up as high as 1,000,000. The Developer Portal containers take **`inotify`** watches only when they need them. The full number is not reserved or held, so it is acceptable to set this value high.

About this task

You must deploy the Developer Portal OVA template to create each Developer Portal node that you want in your cloud. Each node has a separate CLI password account that is required to log in through a Secure Shell (SSH) to complete specific administrative actions for only that node.

Important deployment information for Developer Portal:

- You must deploy the Developer Portal OVA template by using a version of the VMware vSphere Client that supports the SHA-512 Cryptographic Hash Algorithm.
- The Developer Portal node is initially configured with a default password of `7iron-hide`, with the user name of `admin`. For security reasons, change the default password by completing one the following actions:
  - During deployment, if the feature is available in your VMware instance, enter a new password. If you specify a password during deployment, the password for the Developer Portal command line interface (CLI) is modified. Use the new password when you log into the CLI. You cannot modify the admin user name for the CLI.
  - After deployment, log in to the CLI for each virtual appliance and run the command to change the default password for that specific node. The CLI command is **passwd**.

  Note that this console uses a US keyboard configuration, in which the `@` symbol can be in a different place to other keyboard configurations. If you are not using a US keyboard, ensure that you typed the password correctly.
- Only static IP addresses that are specified during the **`apicup`** project configuration before the installation of the OVAs are supported.
- To enable effective high availability for your Portal service, you need a latency that is less than 50ms between all OVAs to avoid the risk of performance degradation. Servers with uniform specifications are required, as any write actions occur at the speed of the slowest OVA, as the write actions are synchronous across the cluster of OVAs. It is recommended that there are three servers in each cluster of OVAs for the high availability configuration. The three servers can be situated in the same data center, or across three data centers to ensure the best availability. However, you can configure high availability with two data centers.

Procedure

Complete the following steps in order:

Note: The following steps apply to VMware only. Depending on the VMware version that you are using, some of the steps might vary. For example, you might not be able to change the user name and password during deployment.

1. Specify configuration settings for the Developer Portal subsystem and create an ISO. See Configuring the Developer Portal subsystem
2. Deploy the subsystem OVA into an appliance VM. See Deploying the Developer Portal subsystem OVA file.
3. Verify the deployment. See Verifying deployment of the Developer Portal subsystem
4. Optional: Define multiple Developer Portal endpoints. See Defining multiple portal endpoints for a VMware environment

What to do next

If you want to deploy an API Connect Analytics OVA file, continue with Deploying the Analytics virtual appliance.

If you did not specify a new password during deployment in VMware, then after deployment, log in to the command-line interface (CLI) for each appliance and run the command **passwd** to change the password.

- **Configuring the Developer Portal subsystem**
  Specify configuration properties for your Developer Portal subystem, and create an ISO.
- **Deploying the Developer Portal subsystem OVA file**
  After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.
- **Verifying deployment of the Developer Portal subsystem**
  Verify that deployment of the Developer Portal OVA file succeeded.
- **Configuring two NICs on the Developer Portal**
  Configure the Developer Portal to use two network interface cards.
- **Specifying a range of allowable client IP addresses for Developer Portal**
  Specify the range of client IP addresses to be allowed in a deployment of the Developer Portal
- **Defining multiple portal endpoints for a VMware environment**
  Multiple public facing endpoints (`portal-www`) can be defined for the Developer Portal.

Configuring the Developer Portal subsystem

Specify configuration properties for your Developer Portal subystem, and create an ISO.

Before you begin

Review the requirements and important deployment information for Developer Portal. See Deploying the Developer Portal virtual appliance.

About this task

Use the `apicup` installation utility to specify configuration settings for your Developer Portal subsystem.

Procedure

1. Ensure that you obtained the distribution file and have a project directory, as described in First steps for deploying in a VMware environment.
2. Change to the project directory.

   ```
   cd myProject
   ```

3. Create a portal subsystem.

   ```
   apicup subsys create port portal
   ```

   Where:
   - *port* is the name of the Developer Portal server that you are creating. You can assign it any name, as long as the identifier consists of lowercase alphanumeric characters or '-', with no spaces, starts with an alphabetic character, and ends with an alphanumeric character.
   - `portal` indicates that you want it to create a Developer Portal microservice.

   Note: Unless you are installing a two data center deployment, leave `multi-site-ha-enabled` as false, and ignore `multi-site-ha-mode`, `replication-peer-fqdn`, `site-name`, and the `portal-replication` endpoint, as these options are all related to that feature. For more information, see Installing a two data center deployment.
   The apiconnect-up-v10.yml file that is in that directory is updated to add the portal-related entries.
   Tip: At any time, you can view the current developer portal subsystem values in the apiconnect-up-v10.yml by running the `apicup get` command:

   ```
   apicup subsys get port
   ```

   If you have not yet configured the subsystem, the command might return errors. Also, if you have not updated the value, a default value is listed, if there is one that is available.
   After configuration is complete, you can view output similar to the following sample:

   ```
   Appliance settings
   ==================

   Name                          Value
   Description
   ----                          -----
   ------
   additional-cloud-init-file
   ```

```
(Optional) Path to additional cloud-init yml file
data-device                    sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld5l8TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/
(Optional) Console login password for `apicadm` user, password must be pre-hashed
dns-servers                    [192.168.1.201]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network                172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network            172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                   eth0
Device for API/UI traffic (Eg: eth0)
search-domain                  [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                   [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                  eth0
Device for cluster traffic (Eg: eth0)


Subsystem settings
==================

Name                           Value
Description
----                           -----
------
deployment-profile             n1xc2.m8
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for
portal
license-use                    nonproduction
License use (production/nonproduction)
multi-site-ha-enabled          false
Multi site HA enabled
multi-site-ha-mode             active
Multi site HA mode (active/passive)
replication-peer-fqdn
Replication peer fully qualified name (replication endpoint of active mode site)
site-name
Site name, used in k8s resource names


Endpoints
=========

Name                           Value
Description
----                           -----
------
portal-admin                   portal-api.example.com
FQDN of Portal admin endpoint
portal-replication
FQDN of Portal replication endpoint, only required if HA is enabled
portal-www                     portal-www.example.com
FQDN of Portal web endpoint
```

4. Specify `deployment-profile`.

   ```
   apicup subsys set port deployment-profile=n3xc4.m8
   ```

   The **deployment-profile** parameter indicates that you are deploying a three replica or one replica environment. For more information, see Planning your deployment topology and profiles.

   Note: The deployment profiles that are shown in the **Description** column of the **apicup get** output are not correct. The available profiles are documented in Planning your deployment topology and profiles.
5. Specify the license version you purchased.
   ```
   apicup subsys set port
   license-use=<license_type>
   ```

   The *license_type* must be either **production** or **nonproduction**. If not specified, the default value is **nonproduction**.

6. Optional: Configure scheduled backups of the subsystem. This step is optional but is recommended.
   Refer to the instructions in Backing up and restoring the Developer Portal.

7. Optional: Configure your logging.
   Logging can be configured at a later time, but you must enable it before installation to capture the log events from the installation.
   a. Complete the procedure at Configuring remote logging for a VMware deployment.
   b. Enter the following command to create the log file:

      ```
      apicup subsys set port additional-cloud-init-file=config_file.yml
      ```

8. Enter the following commands to update the apiconnect-up-v10.yml with the information for your environment:
   a. Use **apicup** to set your endpoints.
      You can use wildcard aliases or host aliases with your endpoints.
      Optionally, you can specify all endpoints with one **apicup** command.

      Note: You cannot specify the underscore character "_" in domain names that are used in endpoints. See API Connect configuration on VMware.
      The endpoints must be unique hostnames which both point to the IP address of the OVA (single node deployment), or to the IP of a load balancer configured in front of the OVA nodes. See examples in the sample output in step 3.

Setting	Endpoint host description
portal-admin	This is the *unique_hostname* for communication between your Cloud Manager and API Manager, and your Developer Portal. The values for the portal-admin and portal-www must be different. `apicup subsys set port portal-admin=unique_hostname.domain`
portal-www	This is the *unique_hostname* for the Developer Portal Internet site that is created for the Developer Portal. Multiple portal-www endpoints may be configured, as described here: Defining multiple portal endpoints for a VMware environment. `apicup subsys set port portal-www=unique_hostname.domain`

b. Set your search domain. Multiple search domains should be separated by commas.

```
apicup subsys set port search-domain=your_search_domain
```

Where *your_search_domain* is the domain of your servers, entered in all lowercase. Setting this value ensures that your searches also append these values, which are based on your company's DNS resolution, at the end of the search value. A sample search domain is *mycompany.example.com*.

Ensure that the value for *your_search_domain* is resolved in the system's /etc/resolv.conf file to avoid "502" errors when accessing the Cloud Manager web site. For example:

```
# Generated by resolvconf
search your_search_domain ibm.com other.domain.com
```

c. Set your domain name servers (DNS).
Supply the IP addresses of the DNS servers for your network. Use a comma to separate multiple server addresses.

```
apicup subsys set port dns-servers=ip_address_of_dns_server[,ip_address_of_another_dns_server_if_necessary]
```

DNS entries may not be changed on a cluster after the initial installation.

9. Set a Public key.

```
apicup subsys set port ssh-keyfiles=path_to_public_ssh_keyfile
```

Setting this key enables you to use **ssh** with this key to log in to the virtual machine to check the status of the installation. You will perform this check in Verifying deployment of the Developer Portal subsystem.

10. If there are multiple Management subsystems in the same project, set the Portal subsystem's `platform-api` and `consumer-api` certificates to match those used by the appropriate Management subsystem to ensure that the Portal subsystem is correctly associated with that Management subsystem.
This step only applies if you installed more than one Management subsystem into a single project.

A Portal subsystem can be associated with only one Management subsystem. To associate the new Portal subsystem with the appropriate Management subsystem, manually set the `mgmt-platform-api` and `mgmt-consumer-api` certificates to match the ones used by the Management subsystem.

a. Run the following commands to get the certificates from the Management subsystem:

```
apicup certs get mgmt-subsystem-name platform-api -t cert > platform-api.crt
apicup certs get mgmt-subsystem-name platform-api -t key > platform-api.key
apicup certs get mgmt-subsystem-name platform-api -t ca > platform-api-ca.crt

apicup certs get mgmt-subsystem-name consumer-api -t cert > consumer-api.crt
apicup certs get mgmt-subsystem-name consumer-api -t key > consumer-api.key
apicup certs get mgmt-subsystem-name consumer-api -t ca > consumer-api-ca.crt
```

where *mgmt-subsystem-name* is the name of the specific Management subsystem that you want to associate the new Portal subsystem with.

b. Run the following commands to set the Portal's certificates to match those used by the Management subsystem:

```
apicup certs set ptl-subsystem-name mgmt-platform-api Cert_file_path Key_file_path CA_file_path

apicup certs set ptl-subsystem-name mgmt-consumer-api Cert_file_path Key_file_path CA_file_path
```

11. You can set a hashed password that you enter to log in to your Developer Portal server for the first time.
a. **Important**: Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.
b. If you do not have a password hashing utility, install one.

Operating system	Command
Ubuntu, Debian, OSX	If the **mkpasswd** command utility is not available, download and install it. (You can also use a different password hashing utility.) On OSX, use the command: `gem install mkpasswd` .
Windows, Red Hat	If necessary, a password hashing utility for the Windows operating system, like OpenSSL

c. Create a hashed password

Operating system	Command
Ubuntu, Debian, OSX	`mkpasswd --method=sha-512 --rounds=4096 password`
Windows, Red Hat	For example, using OpenSSL: `openssl passwd -1 password`. Note that you might need to add your password hashing utility to your path; for example, on Windows: `set PATH=c:\cygwin64\bin;%PATH%`

d. Set the hashed password for your subsystem:

```
apicup subsys set port default-password="hashed_password"
```

Notes:

- The password is hashed. If it is in plain text, you cannot log into the VMWare console.

- Note that the password can only be used to login through the VMware console. You cannot use it to SSH into the Appliance as an alternative to using the **ssh-keyfiles**.
- On Linux or OSX, use single quotes around *hashed_password*. For Windows, use double quotes.
- If you are using a non-English keyboard, understand the limitations with using the remote VMware console. See Requirements for initial deployment on VMware.

12. Optional: If the default IP ranges for the API Connect Kubernetes pod and the service networks conflict with IP addresses that must be used by other processes in your deployment, modify the API Connect values.
    You can change the IP ranges of the Kubernetes pod and the service networks from the default values of 172.16.0.0/16 and 172.17.0.0/16, respectively. In the case that a /16 subnet overlaps with existing IPs on the network, a Classless Inter-Domain Routing (CIDR) as small as /22 is acceptable. You can modify these ranges during initial installation and configuration only. You cannot modify them once an appliance has been deployed. See API Connect configuration on VMware.
    a. Update the IP range for the Kubernetes pod

       ```
       apicup subsys set port k8s-pod-network='new_pod_range'
       ```

       Where *new_pod_range* is the new value for the range.

    b. Update the IP range for Service networks.

       ```
       apicup subsys set port k8s-service-network='new_service_range'
       ```

       Where *new_service _range* is the new value for the range.

13. Add your hosts.

    ```
    apicup hosts create port hostname.domainname hd_password
    ```

    Where the following are true:
    - *hostname.domainname* is the fully qualified name of the server where you are hosting your Developer Portal, including the domain information.
    - *hd_password* is the password of the Linux Unified Key Setup uses to encrypt the storage for your Developer Portal. This password is hashed when it is stored on the server or in the ISO. Note that the password is base64 encoded when stored in **apiconnect-up-v10.yml**.
      Repeat this command for each host that you want to add.

      Note: Host names and DNS entries may not be changed on a cluster after the initial installation.

14. Create your interfaces.

    ```
    apicup iface create port hostname.domainname physical_network_id host_ip_address/subnet_mask gateway_ip_address
    ```

    Where *physical_network_id* is the network interface ID of your physical server. The value is most often **eth0**. The value can also be **ethx**, where **x** is a number identifier.
    The format is similar to this example: **apicup iface create port myHostname.domain eth0 192.0.2.1/255.255.1.1 192.0.2.1**

    Note: You can optionally configure a second network interface (NIC) card for use with Developer Portal. In this scenario, one NIC is used for internal traffic between the Portal and APIM, and the second is used as a public interface. If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway. For information on configuring two NICs for Developer Portal, see Configuring two NICs on the Developer Portal.

15. Optional: You can specify a range of allowable client IP addresses.
    You can optionally restrict which IP addresses can access any of the ingresses by creating an allowlist annotation, which then will only allow the specified IP addresses to access the ingress and will deny all other source IP addresses. See Specifying a range of allowable client IP addresses for Developer Portal.

    Note: One usage of this restriction is the scenario where you split traffic between your private and public networks in 2 NIC configuration.

16. Optional: Use **apicup** to view the configured hosts:

    ```
    apicup hosts list port
    apimdev0232.hursley.ibm.com
     Device  IP/Mask                    Gateway
     eth0    1.2.152.232/255.255.254.0  1.2.152.1
    ```

    Note: If you are configuring a two data center deployment, continue with the installation instructions in Installing a two data center deployment.

17. Optional: Enable JWT security instead of mTLS for communication from management to your portal subsystem.
    JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see Enable JWT instead of mTLS. To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

    ```
    apicup subsys get <management subsystem>

    ...
    jwks-url     https://appliance1.apic.acme.com/api/cloud/oauth2/certs  JWKS URL for Portal and analytics subsystems to
    validate JWT -- this is unsettable and is generated based on the platform-api endpoint
    ...
    ```

    Disable mTLS and enable JWT by setting the jwks-url with **apicup**:

    ```
    apicup subsys set port mtls-validate-client=false
    apicup subsys set port jwks-url=https://appliance1.apic.acme.com/api/cloud/oauth2/certs
    ```

    Note: Do not disable mTLS without enabling JWT.

18. Leave **multi-site-ha-enabled** set to **false**.
19. Optional: Verify that the configuration settings are valid.

    ```
    apicup subsys get port --validate
    ```

    The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting. See the following sample output.

    ```
    Appliance settings
    ==================

    Name                          Value
    ```

```
----                            -----
additional-cloud-init-file
✔
data-device                     sdb
✔
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld5l8TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/   ✔
dns-servers                     [192.168.1.201]
✔
extra-values-file
✔
k8s-pod-network                 172.16.0.0/16
✔
k8s-service-network             172.17.0.0/16
✔
public-iface                    eth0
✔
search-domain                   [subnet1.example.com]
✔
ssh-keyfiles                    [/home/vsphere/.ssh/id_rsa.pub]
✔
traffic-iface                   eth0
✔


Subsystem settings
==================

Name                            Value
----                            -----
deployment-profile              n1xc2.m8
✔
license-use                     nonproduction
✔
multi-site-ha-enabled           false
✔
multi-site-ha-mode              active
✔
replication-peer-fqdn
✔
site-name
✔


Endpoints
=========

Name                            Value
----                            -----
portal-admin                    portal-api.example.com
✔
portal-replication
✔
portal-www                      portal-www.example.com
✔
```

20. Create your ISO file.

    ```
    apicup subsys install port --out portplan-out
    ```

    The **--out** parameter and value are required. In this example, the ISO file is created in the myProject/portplan-out directory.

    If the system cannot find the path to your software that creates ISO files, create a path setting to that software by running a command similar to the following command:

Operating system	Command
Ubuntu, Debian, OSX	`export PATH=$PATH:/Users/your_path/`
Windows, Red Hat	`set PATH="c:\Program Files (x86)\cdrtools";%PATH%`

21. Next, use the ISO to deploy the subsystem OVA. Continue with Deploying the Developer Portal subsystem OVA file.

Deploying the Developer Portal subsystem OVA file

After you have specified configuration settings and created an ISO, deploy the subsystem OVA into an appliance VM.

Before you begin

You must have completed the configuration of the project file entries for the Developer Portal subsystem, in Configuring the Developer Portal subsystem.

Procedure

1. Log into the VMWare vSphere Web Client.
2. Using the VSphere Navigator, navigate to the directory where you are deploying the OVA file.
3. Right-click the directory and select Deploy OVF Template.
4. Complete the Deploy OVF Template wizard.

a. Select the Portal subsystem .OVA file template by navigating to the location where you downloaded the file in First steps for deploying in a VMware environment.
b. Enter a name and location for your file.
c. Select a resource for your template.
d. Review the details for your template.
e. Select the size of your configuration.
f. Select the storage settings.
Note that the number of Central Processing Units, RAM, and size of disk that you need for the Developer Portal varies depending on the number of sites that are hosted and the number of concurrent users you expect your site to have:

Table 1. Developer Portal hardware requirements

Number of sites	Number of concurrent users	Number of CPUs	Amount of RAM (GB)	Data Disk Size (GB)++
1	1	2**	4	50
20	5	4	16	70
100	20	8	32	100
100	100	16	64	500

Important:
- ++The data disk size is extra to the main disk of the OVA. The main disk of the OVA is sized at 100GB, and should not be changed. Therefore, the total disk size of the OVA is 100GB plus the data disk size. The default data disk size is 200GB. If you want to select a different value, then you need to do this by using the OVF Tool, or the VMware GUI, before you power on the VM. Note that certain versions of the VMware GUI may not allow you to resize the data disk, and in this case you should use the OVF Tool.
- **The requirement of 2 CPUs is suitable only for proof-of-concept work, and non-high availability deployments. For example, this configuration is suitable for the demo mode of the apicup installation, which is set by using the command `apicup subsys set` *`port`* `deployment-profile=n1xc2.m8`.
- It's not recommended to have more than 100 sites per Developer Portal service. Note that it's not necessary to have a Portal site for every Catalog, for example Catalogs that are only for API Developers don't need a Portal site, as the APIs can be tested by using credentials from the API Manager. If more than 100 sites are required, you should configure additional Developer Portal services; see Registering a Portal service.

g. Select the networks.
h. Customize the Template, if necessary.
i. Review the details to ensure that they are correct.
j. Select Finish to deploy the virtual machine.

Note: Do not change the OVA hardware version, even if the VMware UI shows a Compatibility range that includes other versions. See Requirements for initial deployment on VMware.
The template creation appears in your Recent Tasks list.
5. Select the Storage tab in the Navigator.
6. Navigate to your datastore.
7. Upload your ISO file.
a. Select the Navigate to the datastore file browser icon in the icon menu.
b. Select the Upload a file to the Datastore icon in the icon menu.
c. Navigate to the ISO file that you created in your project.
It is the myProject/*port*plan-out
d. Upload the ISO file to the datastore.
8. Leave the datastore by selecting the VMs and Templates icon in the Navigator.
9. Locate and select your virtual machine.
10. Select the Configure tab in the main window.
11. Select Edit....
a. On the Virtual Hardware tab, select CD/DVD Drive 1.
b. For the Client Device, select Datastore ISO File.
c. Find and select your datastore in the Datastores category.
d. Find and select your ISO file in the Contents category.
e. Select OK to commit your selection and exit the Select File window.
f. Ensure that the Connect At Power On check box is selected.
Tip:
- Expand the CD/DVD drive 1 entry to view the details and the complete Connect At Power On label.
- Note that VMware related issues with ISO mounting at boot may occur if Connect At Power On

g. Select OK to commit your selection and close the window.
12. Start the virtual machine by selecting the play button on the icon bar.
The installation might take several minutes to complete, depending on the availability of the system and the download speed.
13. Continue with Verifying deployment of the Developer Portal subsystem.

Verifying deployment of the Developer Portal subsystem

Verify that deployment of the Developer Portal OVA file succeeded.

Before you begin

You must have completed Deploying the Developer Portal subsystem OVA file.

About this task

After you configure an ISO and deploy it with the Developer Portal subsystem OVA, as described in Deploying the Developer Portal subsystem OVA file, verify that the subsystem installed correctly and is functional.

Procedure

1. Log in to the virtual machine by using an SSH tool to check the status of the installation:
   a. Enter the following command to connect to *port* using SSH:

      ```
      ssh ip_address -l apicadm
      ```

      You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.
   b. Select Yes to continue connecting.
      Your host names are automatically added to your list of hosts.
   c. Run the **apic status** command to verify that the installation completed and the system is running correctly.
      The command output for a correctly running Developer Portal system is similar to the following lines:

      ```
      $ sudo apic status
      INFO[0000] Log level: info

      Cluster members:
      - testsrv1251.subnet1.example.com (1.2.3.4)
        Type: BOOTSTRAP_MASTER
        Install stage: DONE
        Upgrade stage: NONE
        Docker status:
          Systemd unit: running
        Kubernetes status:
          Systemd unit: running
          Kubelet version: testsrv1251 (4.4.0-137-generic) [Kubelet v1.10.6, Proxy v1.10.6]
        Etcd status: pod etcd-testsrv1251 in namespace kube-system has status Running
        Addons: calico, dns, helm, kube-proxy, metrics-server, nginx-ingress,
      Etcd cluster state:
      - etcd member name: testsrv1251.subnet1.example.com, member id: 10293853252850049269, cluster id:
      17044377177359475136, leader id: 10293853252850049269, revision: 1485879, version: 3.1.17

      Pods Summary:
      NODE               NAMESPACE          NAME                                                READY
      STATUS          REASON
      testsrv1251        default            re702738954-apic-portal-db-f89vn                    2/2
      Running
                         default            re702738954-apic-portal-nginx-6ffb8676d9-gtfc6      0/0
      Pending
                         default            re702738954-apic-portal-nginx-6ffb8676d9-nqdzf      0/0
      Pending
      testsrv1251        default            re702738954-apic-portal-nginx-6ffb8676d9-q85mt      1/1
      Running
      testsrv1251        default            re702738954-apic-portal-www-p9bvx                   2/2
      Running
      testsrv1251        kube-system        calico-node-xkpbk                                   2/2
      Running
      testsrv1251        kube-system        coredns-87cb95869-p4qhf                             1/1
      Running
      testsrv1251        kube-system        coredns-87cb95869-z2n5z                             1/1
      Running
      testsrv1251        kube-system        etcd-testsrv1251                                    1/1
      Running
      testsrv1251        kube-system        ingress-nginx-ingress-controller-dsnxw              1/1
      Running
      testsrv1251        kube-system        ingress-nginx-ingress-default-backend-6f58fb5f56-ldx7t  1/1
      Running
      testsrv1251        kube-system        kube-apiserver-testsrv1251                          1/1
      Running
      testsrv1251        kube-system        kube-apiserver-proxy-testsrv1251                    1/1
      Running
      testsrv1251        kube-system        kube-controller-manager-testsrv1251                 1/1
      Running
      testsrv1251        kube-system        kube-proxy-4pp8v                                    1/1
      Running
      testsrv1251        kube-system        kube-scheduler-testsrv1251                          1/1
      Running
      testsrv1251        kube-system        metrics-server-6fbfb84cdd-hkztz                     1/1
      Running
      testsrv1251        kube-system        tiller-deploy-84f4c8bb78-v6k95                      1/1
      Running
      ```

2. Optional: Define multiple Developer Portal endpoints. See Defining multiple portal endpoints for a VMware environment.
3. If you have now installed all subsystems, continue to Post-deployment steps.

   Note:
   When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the **portal-www** pod, admin container logs, if the endpoint cannot be reached:

   ```
   An error occurred contacting the provided portal web endpoint: example.com
   The provided Portal web endpoint example.com returned HTTP status code 504
   ```

   In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.
   To disable the endpoint check, complete the following update:

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

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

```
      name: admin
    name: www
```

Run the following commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>
```

```
apicup subsys install <ptl_subsys>
```

Configuring two NICs on the Developer Portal

Configure the Developer Portal to use two network interface cards.

About this task

When deploying the Developer Portal, you can optionally separate network traffic such that the portal has network interface in your private network, where the communications to the APIM manager take places, and another network interface in your public network, where the end users can access the Portal web sites.

You can do this by using the iface on eth0 for your traffic and defining an extra iface definition (eth1) for your public interface. Do this as part of the initial configuration of the Developer Portal. Note the following requirements:

- iface eth0 and eth1 must have different subnets. This means that you cannot test the configuration by using two IP addresses on the same subnet, due to the routing table produced by the OVA at bootup.
- The hostname of the machine must be reachable via eth0 and you must use eth0 as the traffic (private interface) and eth1 as the public interface.

Procedure

1. Verify that you have created a first interface, by following the instructions in Step 13 of Configuring the Developer Portal subsystem.
2. Use `apicup` to add an extra iface definition as follows for your public interface:

   ```
   apicup iface create test-stack-portal test0094.test.example.com eth1 1.2.3.232/255.255.254.0 1.2.3.1
   ```

3. Set the hostname of your portal-www ingress to the hostname that is associated with the IP address given on `eth1`.

   ```
   apicup subsys set my-portal portal-www portal.myhost0232.company.com
   ```

4. When configuring the second NIC during initial installation, return to Step 14 of Configuring the Developer Portal subsystem. Follow the remaining steps in the configuration instructions, and then create the Portal OVA and deploy it as usual.
   Notes:
   - If you are creating multiple network interfaces, each one must be on a different subnet with a different gateway.
   - Before powering on the Portal, use the VMware GUI to add a second network card to the VM, with the correct settings for your public network.

Specifying a range of allowable client IP addresses for Developer Portal

Specify the range of client IP addresses to be allowed in a deployment of the Developer Portal

About this task

Use an extra-values-file, which serves as a portal CR override, to provide an allowable addresses annotation.

Procedure

In your `apicup` project, create a file called `ptl-extra-values.yaml` with the necessary content. For example:

```
spec:
  portalUIEndpoint:
      annotations:
        ingress.kubernetes.io/whitelist-source-range: 1.2.88.0/24
```

In this example, `1.2.88.0/24` is the acceptable range of client IP addresses.

Note: Depending on your network configuration for the forwarding of client IP addresses, you might need to run several test iterations to work out which client IP addresses to include in the list.

Defining multiple portal endpoints for a VMware environment

Multiple public facing endpoints (`portal-www`) can be defined for the Developer Portal.

About this task

You can override the single endpoint definition for `portal-www`, and the associated `portal-www-ingress` TLS certificate, to support multiple `portal-www` endpoints.

For information about the endpoints for the Developer Portal, see Configuring the Developer Portal subsystem.

The task requires accessing your portal virtual appliance and creating Kubernetes secrets. Two methods are available for creating Kubernetes secrets:

- The recommended method uses a tool that exists on the portal appliance VM called cert-manager. To use the cert-manager method, skip step 5.
- If you do not want to use the cert-manager method, you can create your Kubernetes secrets manually. Follow step 5 instead of steps 3 and 4.

Procedure

Create **portal-www** endpoint certificates with cert-manager.

1. Log in to your portal virtual appliance with an SSH client, as **apicadm** user

   ```
   ssh apicadm@<subsystem hostname>
   ```

2. Switch to the root user: **sudo -i**
3. Create a file called ingress-issuer.yaml and paste in the following content:

   ```
   apiVersion: cert-manager.io/v1
   kind: Certificate
   metadata:
     name: ingress-ca
   spec:
     secretName: ingress-ca
     commonName: "ingress-ca"
     usages:
     - digital signature
     - key encipherment
     - cert sign
     isCA: true
     duration: 87600h # 10 years
     renewBefore: 720h # 30 days
     privateKey:
       rotationPolicy: Always
     issuerRef:
       name: selfsigning-issuer
       kind: Issuer
   ---
   apiVersion: cert-manager.io/v1
   kind: Issuer
   metadata:
     name: ingress-issuer
   spec:
     ca:
       secretName: ingress-ca
   ```

   Run **kubectl apply -f ingress-issuer.yaml**.

4. Create cert-manager certificate objects for each **portal-www** endpoint.
   a. Create a file called portal-web-certs.yaml, and paste in a **Certificate** entry for each **portal-www** endpoint. In this example, two endpoints are created, so the example file contains two certificates:

      ```
      apiVersion: cert-manager.io/v1
      kind: Certificate
      metadata:
        labels:
          app.kubernetes.io/instance: portal
          app.kubernetes.io/managed-by: ibm-apiconnect
          app.kubernetes.io/name: web-endpoint
        name: portal-web-host1
        namespace: default
      spec:
        dnsNames:
        - <FQDN of portal web endpoint 1>
        duration: 17520h0m0s
        issuerRef:
          group: cert-manager.io
          kind: Issuer
          name: ingress-issuer
        renewBefore: 720h0m0s
        privateKey:
          rotationPolicy: Always
        secretName: portal-web-host1
        usages:
        - digital signature
        - key encipherment
      ---
      apiVersion: cert-manager.io/v1
      kind: Certificate
      metadata:
        labels:
          app.kubernetes.io/instance: portal
          app.kubernetes.io/managed-by: ibm-apiconnect
          app.kubernetes.io/name: web-endpoint
        name: portal-web-host2
        namespace: default
      spec:
        dnsNames:
        - <FQDN of portal web endpoint 2>
        duration: 17520h0m0s
        issuerRef:
          group: cert-manager.io
          kind: Issuer
          name: ingress-issuer
        renewBefore: 720h0m0s
      ```

```
  privateKey:
    rotationPolicy: Always
  secretName: portal-web-host2
  usages:
  - digital signature
  - key encipherment

---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  labels:
    app.kubernetes.io/instance: portal
    app.kubernetes.io/managed-by: ibm-apiconnect
    app.kubernetes.io/name: web-endpoint
  name: portal-web-host3
  namespace: default
spec:
  dnsNames:
  - <FQDN of portal web endpoint 3>
  duration: 17520h0m0s
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: ingress-issuer
  renewBefore: 720h0m0s
  privateKey:
    rotationPolicy: Always
  secretName: portal-web-host3
  usages:
  - digital signature
  - key encipherment
```

b. Apply the file:

```
kubectl apply -f portal-web-certs.yaml
```

Create **portal-www** without cert-manager.

5. Create Kubernetes secrets with OpenSSL. Follow this step instead of steps 3 and 4.
   To create the secrets and certificates for each **portal-www** endpoint, complete the following steps:
   a. If you are using self-signed certificates, create the TLS secrets for each **portal-www** endpoint by generating the certificates. The following example shows how to generate certificates for each **portal-www** endpoint by using OpenSSL:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ptl.host1-tls.key -out ptl.host1-tls.crt -subj
"/CN=ptl.host1.example.com"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ptl.host2-tls.key -out ptl.host2-tls.crt -subj
"/CN=ptl.host2.example.com"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ptl.host3-tls.key -out ptl.host3-tls.crt -subj
"/CN=ptl.host3.example.com"
```

   b. Store the SSL certificates in a secret:
      i. Copy the certificates to the Portal virtual appliance, for example:

```
scp ptl.host1-tls.key ptl.host1-tls.crt ptl.host2-tls.key ptl.host2-tls.crt ptl.host3-tls.key ptl.host3-tls.crt
apicadm@portal-vm-address
```

      ii. Log in to your portal virtual appliance with an SSH client, as **apicadm** user
      iii. Switch to the root user: **sudo -i**
      iv. Create the Kubernetes secrets:

```
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host1 --key ptl.host1-tls.key --
cert ptl.host1-tls.crt
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host2 --key ptl.host2-tls.key --
cert ptl.host2-tls.crt
kubectl --kubeconfig /etc/kubernetes/admin.conf create secret tls portal-web-host3 --key ptl.host3-tls.key --
cert ptl.host3-tls.crt
```

Add **portal-www** endpoints and secrets to your **ptl-extra-values.yaml** file.

6. Exit from the portal VM, and in your **apicup** project directory, create a file called **ptl-extra-values.yaml** with the content:

```
spec:
    portalUIEndpoint:
        annotations: # Remove if cert-manager not used
          cert-manager.io/issuer: ingress-issuer # Remove if cert-manager not used
        hosts:
        - name: ptl.host1.example.com
          secretName: portal-web-host1
        - name: ptl.host2.example.com
          secretName: portal-web-host2
        - name: ptl.host3.example.com
          secretName: portal-web-host3
```

Note: If you followed step 5 and did not use cert-manager to create your secrets, then delete the **annotations** section, so your **ptl-extra-values.yaml** contains:

```
spec:
    portalUIEndpoint:
        hosts:
        - name: ptl.host1.example.com
          secretName: portal-web-host1
        - name: ptl.host2.example.com
          secretName: portal-web-host2
```

```
        - name: ptl.host3.example.com
          secretName: portal-web-host3
```

Run the commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>
```

```
apicup subsys install <ptl_subsys>
```

7. SSH back into your portal virtual appliance, and confirm that multiple hosts exist for your `ptl-portal-web` ingress:

```
sudo kubectl get ingress
NAME                  HOSTS                                                          ADDRESS   PORTS     AGE
ptl-portal-director   api.ptl.example.com                                                      80, 443   25m
ptl-portal-web        ptl.host1.example.com,ptl.host2.example.com,ptl.host3.example.com        80, 443   25m
```

Deploying DataPower Gateway virtual appliance

API Connect uses IBM® DataPower® Gateway to provide the gateway service.

The instructions in this section describe how to deploy IBM DataPower Gateway from files obtained from Passport Advantage®, and apply to the following API Connect scenarios:

- API Connect deployments for the VMware environment
- API Connect deployments for Kubernetes, with a DataPower Gateway in a *non-Kubernetes* environment.

For these scenarios, continue with Installing DataPower Gateway.

Note: Do not use these instructions if you are deploying API Connect deployments for Kubernetes, with DataPower Gateway in a *Kubernetes* environment. In this case, you can use DataPower Gateway files that are packaged with API Connect, on the API Connect section of Fix Central, and configured by using the API Connect `apicup` commands. Follow the instructions in Obtaining product files.

- **Installing DataPower Gateway**
  Install and configure IBM DataPower Gateway in a non-Kubernetes environment for use with API Connect.
- **Configuring DataPower Gateway for API Connect**
  You can configure the IBM DataPower Gateway to prepare for a registration with the API Connect Management server.
- **Sample configuration for multiple peering objects on gateway services external to Kubernetes**
  Sample for reconfiguring the API Connect domain configuration for your gateway services to include multiple peering objects.

Installing DataPower Gateway

Install and configure IBM® DataPower® Gateway in a non-Kubernetes environment for use with API Connect.

Before you begin

Do not use these instructions for either of the following deployment scenarios:

- Both API Connect and DataPower Gateway in a Kubernetes environment.
  In this scenario, use the instructions in Obtaining product files.
- An existing DataPower Gateway in a non-Kubernetes environment, and you want to install a newer Fix Pack.
  In this scenario, complete an *upgrade* rather than a new installation. See Upgrading DataPower Gateway.

To review the installation and configuration scenarios, see Deploying DataPower Gateway virtual appliance.

Procedure

1. Ensure that the DataPower Gateway firmware version you plan to install is compatible with the API Connect management subsystem version.
   Note:
   You can use any combination of API Connect CD releases (10.0.6.x) with DataPower CD releases (DataPower Gateway 10.5.1.x or DataPower API Gateway 10.5.1.x).

   Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

2. Obtain your DataPower files from IBM Passport Advantage®.
   Note that the DataPower files from Passport Advantage have entitled services for API Connect users.

   See also Download the firmware image from Passport Advantage.

3. Follow the DataPower Gateway installation information: https://www.ibm.com/docs/en/datapower-gateway.
   Note: Set the timezone to UTC on all DataPower installations for use with API Connect.
4. Continue with Configuring DataPower Gateway for API Connect.

Configuring DataPower Gateway for API Connect

You can configure the IBM® DataPower® Gateway to prepare for a registration with the API Connect Management server.

About this task

API Connect and DataPower supports two different types of gateway configurations. The DataPower Gateway (v5 compatible) provides the same support as the gateway support that was available with API Connect version 5.x. The DataPower API Gateway is an enhanced gateway that is performance-focused. See API Connect gateway types for more information about the differences between the gateway types.
Continue with the instructions for the type of Gateway Service you are configuring:

- **Configuring DataPower API Gateway**
  Configure the DataPower Gateway to prepare for registration with the API Connect Management server.
- **Configuring DataPower Gateway (v5 compatible)**
  You can configure the DataPower Gateway (v5 compatible) to prepare for a registration with the API Connect Management server.

Configuring DataPower API Gateway

Configure the DataPower Gateway to prepare for registration with the API Connect Management server.

Before you begin

- These instructions are for DataPower® API Gateway deployments in a non-Kubernetes environment. Do not use these instructions if you installed API Connect on Kubernetes, with your DataPower API Gateway Service also in the Kubernetes environment. To review deployment scenarios, see Deploying DataPower Gateway virtual appliance.
- Ensure you have installed a version of DataPower API Gateway that is compatible with the version of the API Connect Management server. See Installing DataPower Gateway.
- A shared certificate and private key is used for securing the communication between the API Connect Management server and the gateway. See *Generating keys and certificates* in the appropriate version of the DataPower documentation for instructions on how to create them with the DataPower tools.
- Ensure that the time zone for the DataPower API Gateway is set to UTC.

About this task

- These instructions provide the basic steps for configuring a gateway service with a single gateway server. The lowest-level configuration objects are created first, then used in other configuration objects.
- Adding gateways to configure a peering environment is similar to creating the first gateway, and is recommended for resiliency in a production environment. A minimum of three gateway servers in a gateway service is recommended for high availability. See *Gateway peering* in the DataPower documentation for more information about configuring additional gateways for peering. See *Providing the gateway service to API Connect* in the DataPower documentation for more details about the DataPower settings and procedures.

Procedure

To configure a DataPower API Gateway to communicate with API Connect, complete the following steps:

Note: Use these instructions only for DataPower API Gateway . If you are configuring DataPower Gateway (v5 compatible) see Configuring DataPower Gateway (v5 compatible).

1. Open the DataPower WebGUI interface.
   Most of the configuration procedure is done in the DataPower WebGUI interface, not in the Blueprint Console.
2. Optional: Enable the XML management interface in the default domain.
   The XML management interface is optional for DataPower API Gateway. If enabled, this interface allows you to send status and configuration requests to the DataPower appliance through a standard SOAP interface, using SOAP messages.
   a. Search for XML management interface in the navigation search bar, and select it.
   b. Set the Administrative state to enabled.
   c. You can specify a different port number if you do not want to use the default of **5550**.
   d. Select Apply to make the changes
   e. Save changes to the default domain by selecting Save Configuration.
3. Optional: Enable the REST management interface in the default domain.
   The REST management interface is required if you want to enable to the Trace feature in the assembly Test tab in the API Manager.
   a. Search for REST management interface in the navigation search bar, and select it.
   b. Set the Administrative state to enabled.
   c. You can specify a different port number if you do not want to use the default of **5554**.
   d. Select Apply to make the changes
   e. Save changes to the default domain by selecting Save Configuration.
4. Create an application domain.
   This domain receives your traffic. The name of the DataPower domain where you configure the API Connect Gateway Service must be the same on each DataPower Gateway.
   a. Search for Application domain in the navigation search bar, and select it.
   b. Select Add to create the application domain.
   c. Enter a unique name for your domain.
   d. Ensure that enabled is selected for the Administrative state.
   e. Ensure that the default domain is listed in the Visible application domain list.
   f. Select Apply.
   g. Change to your new application domain by selecting Domain in the menu bar, and selecting the domain that you created.
   h. Select Save changes and switch domains.
   All of the remaining steps on the DataPower gateway must be done in the application domain that you created.

i. Save changes to the domain by selecting Save Configuration.

5. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.
   You must configure the NTP service, the API Connect Gateway Service cannot run without it. See *Managing the NTP service* in the DataPower documentation.

6. Ensure that you have set a unique **`System Identifier`** for each v10 DataPower gateway. See *Initializing the DataPower Gateway* in the DataPower documentation.
7. Create a self-signed certificate and private key to be used to protect the traffic between the management server and the API gateway service process. You can generate a certificate and private key using DataPower or by using other tools, such as **OpenSSL**. See *Generating keys and certificates* in the appropriate version of the DataPower documentation for instructions on how to create a crypto key with the DataPower tools.
8. Upload your private crypto key file to the domain.
   a. Search for Crypto key in the navigation search bar, and select it.
   b. Select Add to create a key object.
   c. Create a unique name for the key object in the Name field.
   d. Select Upload....
   e. Browse for the key file (which must be a .pem or .p12 file) and select it.
   f. If you want to rename it, enter a new name for the file.
   g. Select Upload to move it to the server in the cert:// folder.
   h. Select Apply to save the changes.
9. Upload your crypto certificate file to the domain.
   Note: If your certificate is signed by an Intermediate CA, you must include the entire chain in a single key file (either .pem or .p12) for uploading.
   a. Search for Crypto certificate in the navigation search bar, and select it.
   b. Select Add to create a certificate object.
   c. Create a unique name for the certificate object in the Name field.
   d. Select Upload....
   e. Browse for the key file (which must be a .pem or .p12 file) and select it.
   f. If you want to rename it, enter a new name for the file.
   g. Select Upload to move it to the server in the cert:// folder.
   h. Select Apply to save the changes.
10. Associate the Crypto key with the Crypto certificate by setting the Identification credential.
    a. Search for Crypto Identification Credentials in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a name for your credential.
    d. Ensure that the Administrative state has a value of enabled.
    e. In the Crypto Key field, select the name of the key object that you created from the drop-down menu.
    f. In the Certificate object field, select the name of the certificate object that you created from the drop-down menu.
    g. Select Apply to commit your changes.
11. Create your TLS Client profile.
    a. Search for TLS Client profile in the navigation search bar, and select it.
    b. Select Add to create a client profile.
    c. Create a unique name for the profile in the Name field.
    d. Select your Identification credential from the drop-down list.
    e. Ensure that the value of Validate server certificate is set to off.
    f. Ensure that the value of Use SNI is set to on.
    g. Select Apply to save the changes.
12. Create your TLS Server profile.
    The TLS Server profile secures your gateway director endpoint. When you register your gateway service in the Cloud Manager UI, the gateway returns the server certificate that you configure in this TLS Server profile.
    a. Search for TLS Server Profile in the navigation search bar, and select it.
    b. Select Add to create a server profile.
    c. Create a unique name for the profile in the Name field.
    d. Select your Identification credential from the drop-down list.
    e. Select Apply to save the changes.
    Important: Do not disable Request client authentication unless you enable JWT instead of mTLS. Client authentication is essential for secure mTLS communication between your management and gateway subsystems.
13. Define a configuration sequence.
    The configuration sequence specifies how DataPower implements the APIs that are defined in API Connect.

    a. Search for Configuration sequence in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a name for your configuration sequence.
       The name **`apic-config`** is not allowed because it is already used internally.

    d. Ensure that the Administrative state has a value of enabled.
    e. Ensure that the value in the Location profiles field is set to local:///
       This is the default value, so you might not need to change it.
    f. Select the Access profile.
       For instructions on creating access profiles, see *Configuring the access profile for a configuration sequence* in the appropriate version of the DataPower documentation.

    g. Change the value of the Configuration execution interval field to `3000`.
       The other fields can retain their default settings.
    h. Select Apply to commit your changes.
14. Configure your gateway peering object for the API Connect Gateway Service.
    This step is required when you set up a peer group of gateways, even if there is only a single gateway server in the gateway service.
    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       You can use the default value of 16380.

g. Select a monitor port for the communication.
   You can use the default value of 26380.
h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
i. Clear the Enable TLS check box. TLS is not needed for a single peer.
j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
k. Select Apply to commit your changes.

15. Configure your gateway peering object for rate limit information.
    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       Use a unique port, different than the ports used for communication by other gateway peering objects.
    g. Select a monitor port for the communication.
       Use a unique port, different than the ports used for monitoring by other gateway peering objects.
    h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
    i. Clear the Enable TLS check box. TLS is not needed for a single peer.
    j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
    k. Select Apply to commit your changes.
16. Optional: If you are using the ratelimit module described in the DataPower documentation, configure your gateway peering object for GatewayScript rate limiting information.
    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       Use a unique port, different than the ports used for communication by other gateway peering objects.
    g. Select a monitor port for the communication.
       Use a unique port, different than the ports used for monitoring by other gateway peering objects.
    h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
    i. Clear the Enable TLS check box. TLS is not needed for a single peer.
    j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
    k. Select Apply to commit your changes.
17. Configure your gateway peering object for subscription information.
    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       Use a unique port, different than the ports used for communication by other gateway peering objects.
    g. Select a monitor port for the communication.
       Use a unique port, different than the ports used for monitoring by other gateway peering objects.
    h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
    i. Clear the Enable TLS check box. TLS is not needed for a single peer.
    j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
    k. Select Apply to commit your changes.
18. Configure the gateway peering object for the API probe.
    In order for API Connect to receive trace data in the Test tab's debugger, the DataPower API Gateway must be configured to support the API probe.

    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       Use a unique port, different than the ports used for communication by other gateway peering objects.
    g. Select a monitor port for the communication.
       Use a unique port, different than the ports used for monitoring by other gateway peering objects.
    h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
    i. Clear the Enable TLS check box. TLS is not needed for a single peer.
    j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
    k. Select Apply to commit your changes.
19. Configure the gateway peering manager.
    a. Search for Gateway Peering Manager in the navigation search bar, and select it.
    b. Set the Administrative state to enabled.
    c. In the pull-down menu next to API Connect Gateway Service, select the gateway peering object configured in Step 14 for the API Connect Gateway Service.
    d. In the pull-down menu next to Rate Limit, select the gateway peering object configured in Step 15 for rate limit information.
    e. In the pull-down menu next to Subscription, select the gateway peering object configured in Step 17 for subscription.
    f. In the pull-down menu next to API Probe, select the gateway peering object that you configured in Step 18 for the API probe.
    g. If you are using the ratelimit module described in the DataPower documentation, then for Gateway rate limiting, select the gateway peering object that you configured in Step 16 for rate limit information.
    h. Select Apply to commit your changes.
20. Configure the API probe settings object.
    a. Search for API probe settings in the navigation search bar, and select it.
    b. Set the Administrative state to enabled.
    c. Set the maximum number of records to `1000`.
    d. Set the expiration to `60` minutes.

e. In the pull-down menu next to Gateway Peering, select the gateway peering object that you configured in Step 18 for the API probe.
f. Select Apply to commit your changes.

21. Set the API Connect Gateway service to define the communication interface with the API Connect Management server and for API transactions.
    a. Search for API Connect Gateway service in the navigation search bar, and select it.
    b. Ensure that the Administrative state is set to enabled.
    c. In the Local address field, enter the IP address of the DataPower gateway to which you want the traffic from the API Connect Management server to be sent.
    d. Specify a value for Local port. You can use the default value of 3000, or specify a different port value.
    Note: The Local port specifies the port through which API Connect connects to manage the API Connect Gateway Service. Use this port when you configure a Gateway Service on API Connect. Beyond this port, the gateway service uses two additional consecutive ports after the defined local port to bind to a loopback address. Therefore, you must ensure that there are no conflicts on all three consecutive ports that start from the defined local port.
    e. In the TLS client field drop-down list, select the name of the TLS client profile that you created.
    f. In the TLS server field drop-down list, select the name of the TLS server profile that you created.
    g. In the API gateway address field, enter the IP address for the DataPower gateway to which you want the API traffic sent.
    h. Use the default port value of 9443 for the API gateway port.
    If the port is not being used by another service, you can also change it to port 443 if you want API transactions to be sent to the default port for HTTPS.
    i. For DataPower API Gateway, set the Gateway Peering to **(none)**.
    When no gateway peering object is configured for the DataPower API Gateway, the peering configuration defined in the Gateway Peering Manager configuration is used.
    j. Select whether you want the DataPower Gateway (v5 compatible) or the DataPower API Gateway.
    When the option is selected, it enables the registration of a DataPower Gateway (v5 compatible) gateway. Clear it to enable a DataPower API Gateway.
22. Optional: Enable JWT security instead of mTLS for communication from management to gateway. JWT security provides application layer security and can be used instead of mTLS when load-balancers are located between subsystems that require TLS termination. For more information about JWT security, see Enable JWT instead of mTLS.
    To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the DataPower documentation. The JWKS URL required to enable JWT security can be found in the management subsystem settings:

```
apicup subsys get <management subsystem>

...
jwks-url     https://appliance1.apic.acme.com/api/cloud/oauth2/certs  JWKS URL for Portal and analytics subsystems to
validate JWT -- this is unsettable and is generated based on the platform-api endpoint
...
```

23. Register the gateway service in the API Connect Cloud Manager console:
    a. Open the API Connect Cloud Manager console.
    b. Navigate to Configure Topology.
    c. Select Register Service.
    d. Select DataPower API Gateway for the DataPower API Gateway.
    e. Add a title, name, and summary for the gateway connection.
    f. Optional: Configure the OAuth Shared Secret.
    This setting allows OAuth tokens to be shared across multiple gateway services.
    g. Enter one of the following values in the API Invocation Endpoint field:
        - IP address of the load balancer for the API transactions
        - IP address or host name of one of the gateways

    For example:**https://192.0.2.0:9443/**
    h. Enter the one of the following values in the Management Endpoint field:
        - IP address of the load balancer for the management server traffic set to port 3000
        - IP address or hostname of one of the gateways

    For example:**https://192.0.2.0:3000/**
24. Select the default TLS Client Profile
25. Optional: Configure Server Name Indication (SNI) profiles.
    SNI profiles allow different TLS certificates to be used for API transaction requests from different host names.
26. Enable logging for the gateway.
    a. Use the CLI to configure the following log target in the API Connect application domain:

```
switch domain <apiconnect_domain>
configure terminal
logging target gwd-log
type file
format text
timestamp zulu
size 50000
local-file logtemp:///gwd-log.log
event apic-gw-service debug
exit
apic-gw-service;admin-state disabled;exit
apic-gw-service;admin-state enabled;exit
write mem
exit
```

    b. Repeat this set for every gateway server in the cluster.

Configuring DataPower Gateway (v5 compatible)

You can configure the DataPower Gateway (v5 compatible) to prepare for a registration with the API Connect Management server.

Before you begin

- These instructions are for DataPower® API Gateway deployments in a non-Kubernetes environment. Do not use these instructions if you installed API Connect on Kubernetes, with your DataPower API Gateway Service also in the Kubernetes environment. To review deployment scenarios, see Deploying DataPower Gateway virtual appliance.

- Ensure you have installed a version of DataPower API Gateway that is compatible with the version of the API Connect Management server. See Installing DataPower Gateway.
- A shared certificate and private key is used for securing the communication between the API Connect Management server and the gateway. See *Generating keys and certificates* in the appropriate version of the DataPower documentation for instructions on how to create them with the DataPower tools.
- Ensure that the time zone for the DataPower API Gateway is set to UTC.

About this task

- These instructions provide the basic steps for configuring a gateway service with a single gateway server. The lowest-level configuration objects are created first, then used in other configuration objects.
- Adding gateways to configure a peering environment is similar to creating the first gateway, and is recommended for resiliency in a production environment. A minimum of three gateway servers in a gateway service is recommended for high availability. See *Gateway peering* in the DataPower documentation for more information about configuring additional gateways for peering. See *Providing the gateway service to API Connect* in the DataPower documentation for more details about the DataPower settings and procedures.

Procedure

To configure a DataPower gateway to communicate with API Connect, complete the following steps:

Note: Use these instructions only for DataPower Gateway (v5 compatible). If you are configuring DataPower API Gateway, see Configuring DataPower API Gateway.

1. Open the DataPower WebGUI interface.
   Most of the configuration procedure is done in the DataPower WebGUI interface, not in the Blueprint Console.
2. Enable the XML management interface in the **default** domain, if required. The XML management interface is required for DataPower Gateway (v5 compatible). The XML management interface is optional for DataPower API Gateway.
   a. Search for **XML management interface** in the navigation search bar, and select it.
   b. Set the Administrative state to enabled.
   c. You can specify a different port number if you do not want to use the default of `5550`.
   d. Select Apply to make the changes
   e. Save changes to the default domain by selecting Save Configuration.
3. Create an application domain.
   This domain receives your traffic. The name of the DataPower domain where you configure the API Connect Gateway Service must be the same on each DataPower Gateway.
   a. Search for Application domain in the navigation search bar, and select it.
   b. Select Add to create the application domain.
   c. Enter a unique name for your domain.
   d. Ensure that enabled is selected for the Administrative state.
   e. Ensure that the default domain is listed in the Visible application domain list.
   f. Select Apply.
   g. Change to your new application domain by selecting Domain in the menu bar, and selecting the domain that you created.
   h. Select Save changes and switch domains.
   All of the remaining steps on the DataPower gateway must be done in the application domain that you created.
   i. Save changes to the domain by selecting Save Configuration.
4. For the DataPower Gateway (v5 compatible) only: Enable statistics in the domain you created for **API Connect**.
   a. Search for and select Statistics settings in the navigation search.
   b. Select enabled for the Administrative state.
   c. Select Apply.
5. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.
   See Managing the NTP service.
6. Ensure that you have set a unique **`System Identifier`** for each v10 DataPower gateway. See Initializing the DataPower Gateway.
7. Create a self-signed certificate and private key to be used to protect the traffic between the management server and the API gateway service process. You can generate a certificate and private key using DataPower or by using other tools, such as **OpenSSL**. See *Generating keys and certificates* in the appropriate version of the DataPower documentation for instructions on how to create a crypto key with the DataPower tools.
8. Upload your private crypto key file to the domain.
   a. Search for Crypto key in the navigation search bar, and select it.
   b. Select Add to create a key object.
   c. Create a unique name for the key object in the Name field.
   d. Select Upload....
   e. Browse for the key file (which must be a .pem or .p12 file) and select it.
   f. If you want to rename it, enter a new name for the file.
   g. Select Upload to move it to the server in the cert:// folder.
   h. Select Apply to save the changes.
9. Upload your crypto certificate file to the domain.
   Note: If your certificate is signed by an Intermediate CA, you must include the entire chain in a single key file (either .pem or .p12) for uploading.
   a. Search for Crypto certificate in the navigation search bar, and select it.
   b. Select Add to create a certificate object.
   c. Create a unique name for the certificate object in the Name field.
   d. Select Upload....
   e. Browse for the key file (which must be a .pem or .p12 file) and select it.
   f. If you want to rename it, enter a new name for the file.
   g. Select Upload to move it to the server in the cert:// folder.
   h. Select Apply to save the changes.
10. Associate the Crypto key with the Crypto certificate by setting the Identification credential.
    a. Search for Crypto Identification Credentials in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a name for your credential.
    d. Ensure that the Administrative state has a value of enabled.
    e. In the Crypto Key field, select the name of the key object that you created from the drop-down menu.
    f. In the Certificate object field, select the name of the certificate object that you created from the drop-down menu.

g. Select Apply to commit your changes.

11. Create your TLS Client profile.
    a. Search for TLS Client profile in the navigation search bar, and select it.
    b. Select Add to create a client profile.
    c. Create a unique name for the profile in the Name field.
    d. Select your Identification credential from the drop-down list.
    e. Ensure that the value of Validate server certificate is set to off.
    f. Ensure that the value of Use SNI is set to on.
    g. Select Apply to save the changes.
12. Create your TLS Server profile.
    The TLS Server profile secures your gateway director endpoint. When you register your gateway service in the Cloud Manager UI, the gateway returns the server certificate that you configure in this TLS Server profile.
    a. Search for TLS Server Profile in the navigation search bar, and select it.
    b. Select Add to create a server profile.
    c. Create a unique name for the profile in the Name field.
    d. Select your Identification credential from the drop-down list.
    e. Select Apply to save the changes.

    Important: Do not disable Request client authentication unless you enable JWT instead of mTLS. Client authentication is essential for secure mTLS communication between your management and gateway subsystems.
13. Configure your gateway peering object for the API Connect Gateway Service.
    This step is required when you set up a peer group of gateways, even if there is only a single gateway server in the gateway service.
    a. Search for Gateway peering in the navigation search bar, and select it.
    b. Select Add.
    c. Enter a unique name for your gateway peering object.
    d. Ensure that the Administrative state has a value of enabled.
    e. Select a local address for the communications among the members of the peer group.
    f. Select a local port for the communication.
       You can use the default value of 16380.
    g. Select a monitor port for the communication.
       You can use the default value of 26380.
    h. Because this procedure uses only one gateway, ensure that Peer group mode is not selected.
    i. Clear the Enable TLS check box. TLS is not needed for a single peer.
    j. Set the Persistence location value to **`Memory`** for either physical DataPower appliance or virtual DataPower appliance.
    k. Select Apply to commit your changes.
14. Set the API Connect Gateway service to define the communication interface with the API Connect Management server and for API transactions.
    a. Search for API Connect Gateway service in the navigation search bar, and select it.
    b. Ensure that the Administrative state is set to enabled.
    c. In the Local address field, enter the IP address of the DataPower gateway to which you want the traffic from the API Connect Management server to be sent.
    d. Use the default port value of 3000 for the Local port.
    e. In the TLS client field drop-down list, select the name of the TLS client profile that you created.
    f. In the TLS server field drop-down list, select the name of the TLS server profile that you created.
    g. In the API gateway address field, enter the IP address for the DataPower gateway to which you want the API traffic sent.
    h. Use the default port value of 9443 for the API gateway port.
       If the port is not being used by another service, you can also change it to port 443 if you want API transactions to be sent to the default port for HTTPS.
    i. For DataPower Gateway (v5 compatible), select the gateway peering object that you created in Step 13.
    j. Select whether you want the DataPower Gateway (v5 compatible) or the DataPower API Gateway.
       When the option is selected, it enables the registration of a DataPower Gateway (v5 compatible) gateway.
15. Optional: Enable JWT security instead of mTLS for communication from management to gateway. JWT security provides application layer security and can be used instead of mTLS when load-balancers are located between subsystems that require TLS termination. For more information about JWT security, see Enable JWT instead of mTLS.
    To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the DataPower documentation. The JWKS URL required to enable JWT security can be found in the management subsystem settings:

    ```
    apicup subsys get <management subsystem>

    ...
    jwks-url     https://appliance1.apic.acme.com/api/cloud/oauth2/certs  JWKS URL for Portal and analytics subsystems to
    validate JWT -- this is unsettable and is generated based on the platform-api endpoint
    ...
    ```

16. Register the gateway service in the API Connect Cloud Manager console:
    a. Open the API Connect Cloud Manager console.
    b. Navigate to Configure Topology.
    c. Select Register Service.
    d. Select DataPower Gateway (v5 compatible) for the gateway that was available in version 5.
    e. Add a title, name, and summary for the gateway connection.
    f. Optional: Configure the OAuth Shared Secret.
       This setting allows OAuth tokens to be shared across multiple gateway services.
    g. Enter one of the following values in the API Invocation Endpoint field:
       - IP address of the load balancer for the API transactions
       - IP address or host name of one of the gateways

       For example:**`https://192.0.2.0:9443/`**
    h. Enter the one of the following values in the Management Endpoint field:
       - IP address of the load balancer for the management server traffic set to port 3000
       - IP address or hostname of one of the gateways

       For example:**`https://192.0.2.0:3000/`**
17. Select the default TLS Client Profile
18. Optional: Configure Server Name Indication (SNI) profiles.
    SNI profiles allow different TLS certificates to be used for API transaction requests from different host names.
19. Enable logging for the gateway.
    a. Use the CLI to configure the following log target in the API Connect application domain:

```
switch domain <apiconnect_domain>
configure terminal
logging target gwd-log
type file
format text
timestamp zulu
size 50000
local-file logtemp:///gwd-log.log
event apic-gw-service debug
exit
apic-gw-service;admin-state disabled;exit
apic-gw-service;admin-state enabled;exit
write mem
exit
```

b. Repeat this set for every gateway server in the cluster.

Sample configuration for multiple peering objects on gateway services external to Kubernetes

Sample for reconfiguring the API Connect domain configuration for your gateway services to include multiple peering objects.

- This reference shows how to configure API Connect domain configuration with gateway-peering configuration. To review domain configuration settings, see Configuring DataPower Gateway for API Connect.
- This sample applies only to gateway services that are deployed external to Kubernetes, on either a physical or virtual appliance. This sample does not apply to gateway services that are deployed on Kubernetes.

The following examples show sample configuration for 3 different gateways, listed here with sample IP addresses of 1.1.1.1, 2.2.2.2, and 3.3.3.3. Note the following:

- The priority should be set differently on each of the three gateways. Set the lowest priority for the gateway that will run as primary.
- **persistence** should be set to **memory** for all configured peering objects.
- Optionally, add SSL configuration.

Configuration on gateway 1 (1.1.1.1)

```
top; configure terminal;
    domain apiconnect; visible default; exit;

    sw apiconnect;

    loglevel debug;
    logging target gwd-log
      type file
      format text
      timestamp syslog
      size 50000
      local-file logtemp:///gwd-log
      event apic-gw-service debug
    exit

    config-sequence "apiconnect"
      location "local:///"
      watch "on"
      delete-unused "on"
      match "(.*)\.cfg$"
      summary "Toolkit Reboot configuration"
      run-sequence-interval 3000
      optimize-for-apic on
    exit

    crypto
      key sth_apic sharedcert:///sth_apic-privkey.cer
      certificate sth_apic sharedcert:///sth_apic-sscert.cer
      idcred sth_apic sth_apic sth_apic
      ssl-client gwd_to_mgmt
        idcred sth_apic
        no validate-server-cert
      exit
      ssl-server gwd_to_mgmt
        idcred sth_apic
        no request-client-auth
        validate-client-cert off
      exit
    exit

    gateway-peering subs
      admin enabled
      local-address 1.1.1.1
      local-port 15222
      monitor-port 26222
      priority 100
      enable-ssl off
      enable-peer-group on
      peer 2.2.2.2
      peer 3.3.3.3
      persistence memory
    exit

    gateway-peering rate-limit
```

```
      admin enabled
      local-address 1.1.1.1
      local-port 15223
      monitor-port 26223
      priority 100
      enable-ssl off
      enable-peer-group on
      peer 2.2.2.2
      peer 3.3.3.3
      persistence memory
    exit

    gateway-peering gwd
      admin enabled
      local-address 1.1.1.1
      local-port 15224
      monitor-port 26224
      priority 100
      enable-ssl off
      enable-peer-group on
      peer 2.2.2.2
      peer 3.3.3.3
      persistence memory
    exit

    gateway-peering probe
      admin enabled
      local-address 1.1.1.1
      local-port 15225
      monitor-port 26225
      priority 100
      enable-ssl off
      enable-peer-group on
      peer 2.2.2.2
      peer 3.3.3.3
      persistence memory
    exit

    gateway-peering gws-rate-limit
      admin enabled
      local-address 1.1.1.1
      local-port 15226
      monitor-port 26226
      priority 100
      enable-ssl off
      enable-peer-group on
      peer 2.2.2.2
      peer 3.3.3.3
      persistence memory
    exit

    gateway-peering-manager
      admin enabled
      apic-gw-service gwd
      rate-limit rate-limit
      subscription subs
      apiprobe probe
      ratelimit-module gws-rate-limit

    apic-gw-service
      admin-state enabled
      local-address 0.0.0.0
      local-port 3000
      api-gw-address 0.0.0.0
      api-gw-port 9443
      v5-compatibility-mode off
      ssl-server gwd_to_mgmt
      ssl-client gwd_to_mgmt
    exit

    write mem
```

Configuration on gateway 2 (2.2.2.2)

```
top; configure terminal;
    domain apiconnect; visible default; exit;

    sw apiconnect;

    loglevel debug;
    logging target gwd-log
      type file
      format text
      timestamp syslog
      size 50000
      local-file logtemp:///gwd-log
      event apic-gw-service debug
    exit

    config-sequence "apiconnect"
      location "local:///"
      watch "on"
      delete-unused "on"
      match "(.*)\.cfg$"
      summary "Toolkit Reboot configuration"
      run-sequence-interval 3000
```

```
    optimize-for-apic on
  exit

  crypto
    key sth_apic sharedcert:///sth_apic-privkey.cer
    certificate sth_apic sharedcert:///sth_apic-sscert.cer
    idcred sth_apic sth_apic sth_apic
    ssl-client gwd_to_mgmt
      idcred sth_apic
      no validate-server-cert
    exit
    ssl-server gwd_to_mgmt
      idcred sth_apic
      no request-client-auth
      validate-client-cert off
    exit
  exit

  gateway-peering subs
    admin enabled
    local-address 2.2.2.2
    local-port 15222
    monitor-port 26222
    priority 105
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 3.3.3.3
    persistence memory
  exit

  gateway-peering rate-limit
    admin enabled
    local-address 2.2.2.2
    local-port 15223
    monitor-port 26223
    priority 105
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 3.3.3.3
    persistence memory
  exit

  gateway-peering gwd
    admin enabled
    local-address 2.2.2.2
    local-port 15224
    monitor-port 26224
    priority 105
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 3.3.3.3
    persistence memory
  exit

  gateway-peering probe
    admin enabled
    local-address 2.2.2.2
    local-port 15225
    monitor-port 26225
    priority 105
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 3.3.3.3
    persistence memory
  exit

  gateway-peering gws-rate-limit
    admin enabled
    local-address 2.2.2.2
    local-port 15226
    monitor-port 26226
    priority 105
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 3.3.3.3
    persistence memory
  exit

  gateway-peering-manager
    admin enabled
    apic-gw-service gwd
    rate-limit rate-limit
    subscription subs
    apiprobe probe
    ratelimit-module gws-rate-limit
  exit

  apic-gw-service
    admin-state enabled
    local-address 0.0.0.0
    local-port 3000
```

```
      api-gw-address 0.0.0.0
      api-gw-port 9443
      v5-compatibility-mode off
      ssl-server gwd_to_mgmt
      ssl-client gwd_to_mgmt
    exit

    write mem
```

Configuration on gateway 3 (3.3.3.3)

```
top; configure terminal;
    domain apiconnect; visible default; exit;

    sw apiconnect;

    loglevel debug;
    logging target gwd-log
      type file
      format text
      timestamp syslog
      size 50000
      local-file logtemp:///gwd-log
      event apic-gw-service debug
    exit

    config-sequence "apiconnect"
      location "local:///"
      watch "on"
      delete-unused "on"
      match "(.*)\.cfg$"
      summary "Toolkit Reboot configuration"
      run-sequence-interval 3000
      optimize-for-apic on
    exit

    crypto
      key sth_apic sharedcert:///sth_apic-privkey.cer
      certificate sth_apic sharedcert:///sth_apic-sscert.cer
      idcred sth_apic sth_apic sth_apic
      ssl-client gwd_to_mgmt
        idcred sth_apic
        no validate-server-cert
      exit
      ssl-server gwd_to_mgmt
        idcred sth_apic
        no request-client-auth
        validate-client-cert off
      exit
    exit

    gateway-peering subs
      admin enabled
      local-address 3.3.3.3
      local-port 15222
      monitor-port 26222
      priority 110
      enable-ssl off
      enable-peer-group on
      peer 1.1.1.1
      peer 2.2.2.2
      persistence memory
    exit

    gateway-peering rate-limit
      admin enabled
      local-address 3.3.3.3
      local-port 15223
      monitor-port 26223
      priority 110
      enable-ssl off
      enable-peer-group on
      peer 1.1.1.1
      peer 2.2.2.2
      persistence memory
    exit

    gateway-peering gwd
      admin enabled
      local-address 3.3.3.3
      local-port 15224
      monitor-port 26224
      priority 110
      enable-ssl off
      enable-peer-group on
      peer 1.1.1.1
      peer 2.2.2.2
      persistence memory
    exit

    gateway-peering probe
      admin enabled
      local-address 3.3.3.3
      local-port 15225
      monitor-port 26225
      priority 110
```

```
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 2.2.2.2
    persistence memory
  exit

  gateway-peering gws-rate-limit
    admin enabled
    local-address 3.3.3.3
    local-port 15226
    monitor-port 26226
    priority 110
    enable-ssl off
    enable-peer-group on
    peer 1.1.1.1
    peer 2.2.2.2
    persistence memory
  exit

  gateway-peering-manager
    admin enabled
    apic-gw-service gwd
    rate-limit rate-limit
    subscription subs
    apiprobe probe
    ratelimit-module gws-rate-limit
  exit

  apic-gw-service
    admin-state enabled
    local-address 0.0.0.0
    local-port 3000
    api-gw-address 0.0.0.0
    api-gw-port 9443
    v5-compatibility-mode off
    ssl-server gwd_to_mgmt
    ssl-client gwd_to_mgmt
  exit

  write mem
```

Post-deployment steps

When all subsystems are deployed, use the admin account to access the Cloud Manager administrative console and begin configuration. Optionally, take VMware snapshots of the VMs.

About this task

When you have deployed all required API Connect components, and all subsystems are running, complete the following:

Procedure

1. If you have not previously accessed the Cloud Manager, verify you can access the API Connect Cloud Manager. Enter the URL in your browser.
   The syntax is `https://<hostname.domain>/admin`. For example:

   ```
   https://cloud-admin-ui.testsrv0231.subnet1.example.com/admin
   ```

   The first time that you access the Cloud Manager user interface, you enter `admin` for the user name and `7iron-hide` for the password. You will be prompted to change the Cloud Administrator password and email address. See Accessing the Cloud Manager user interface.

2. Define your API Connect configuration by using the API Connect Cloud Manager. See Cloud Manager configuration checklist.
3. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see Using VM snapshots for infrastructure backup and disaster recovery for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.
   Important: VM snapshots are not an alternative to the standard API Connect backups. You should have completed an API Connect backup of each subsystem when verifying initial deployment, as advised in:
   - Verify installation of the Management subsystem
   - Verifying deployment of the Developer Portal subsystem
   - Verifying deployment of the Analytics subsystem

Installing a two data center deployment

How to install a two data center disaster recovery (2DCDR) deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see Two data center warm-standby deployment strategy on VMware.

Review the information in Requirements for initial deployment on VMware and, if appropriate, What's new for v10 installation from previous versions.

Familiarize yourself with the instructions in First steps for deploying in a VMware environment. Complete these instructions with the additional steps in this topic that are specific to installing a 2DCDR deployment.

Restriction: It is not possible to use the Automated API behavior testing application (Installing the Automated API behavior testing application) in a 2DCDR configuration.

About this task

Installing API Connect as a 2DCDR deployment is similar to a single installation. The deployment in each data center is effectively an instance of the same API Connect deployment. The following endpoints must be the same on both data centers:

Management subsystem endpoints

- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints

- `portal-admin`
- `portal-www`

To install a 2DCDR deployment, you follow the normal stand-alone API Connect installation steps, but before you generate the ISO files you must set some additional 2DCDR properties.

The 2DCDR properties are described in the following table.

Table 1. 2DCDR properties

Setting	Description
`multi-site-ha-enabled`	Indicates whether 2DCDR deployment is enabled. Set to `true` to enable 2DCDR. Set to `false` to disable 2DCDR. The default value is `false`.
`multi-site-ha-mode`	The 2DCDR mode of the data center. Possible values are: • `active` - indicates the primary data center. • `passive` - indicates the warm-standby data center.
`management-replication`	Required only for the management subsystem. The external ingress name for the management subsystem in the current data center in the 2DCDR deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
`portal-replication`	Required only for the portal subsystem. The external ingress name for the portal subsystem in the current data center in the 2DCDR deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
`replication-peer-fqdn`	The ingress hostname for the other data center in the 2DCDR deployment. This information is required so that the two data centers can communicate with each other.
`siteName`	Unique descriptive name for the API Connect data center. Must be different in each data center. This name is used in the hostnames, and so can contain only `a-z` and `0-9` characters. You can set a `siteName` only when the subsystems are first deployed, and you cannot change the name after deployment. If you don't set `siteName` at first deployment, the `siteName` is autogenerated.

Important:

- Use a single APICUP project for all subsystems so that the API Connect deployments in both data centers have the same certificate chains. If network access to both data centers from the same APICUP project location is not possible, you can copy the project directory across to the other data center. If you are copying the project directory to your other data center, ensure that you keep backups of both project directories.
- The original project directory that is created with APICUP during the initial product installation is required to both restore the database and to upgrade your deployment. You cannot restore the database or complete an upgrade without the initial project directory because it contains pertinent information about the cluster.
- Subsystem endpoints for the components cannot be changed.
- The subsystem name of the management and portal subsystems must consist of lowercase alphanumeric characters or '-', contain no spaces, start with an alphabetic character, and end with an alphanumeric character.

Procedure

- **Installing on the active data center**
  The following example shows how to configure the 2DCDR settings for deploying on the active data center (DC). In this example, the active data center is called `dallas` (DC1), and the warm-standby data center is called `raleigh` (DC2).

  1. Complete the steps in Configuring the Management subsystem up to and including step 14 to configure the standard (non-2DCDR) settings for the management subsystem in Dallas (DC1).
  2. Configure the 2DCDR values to set Dallas to be active for the management subsystem in DC1.
     Run the following commands:

     ```
     apicup subsys set mgmt_dallas multi-site-ha-enabled=true

     apicup subsys set mgmt_dallas multi-site-ha-mode=active

     apicup certs set mgmt_dallas management-replication-ingress --clear

     apicup subsys set mgmt_dallas management-replication=mgrreplicationdallas.cluster1.example.com

     apicup subsys set mgmt_dallas replication-peer-fqdn=mgrreplicationraleigh.cluster2.example.com

     apicup subsys set mgmt_dallas site-name=dallas
     ```

     where
     - *`mgmt_dallas`* is the name of the management subsystem that you are configuring.

- ***mgrreplicationdallas.cluster1.example.com*** is the ingress hostname for the current data center in the two data center deployment.
- ***mgrreplicationraleigh.cluster2.example.com*** is the ingress hostname for the other data center in the two data center deployment.
- ***dallas*** is the name of the site in the current data center.

3. Complete the remainder of the steps in Configuring the Management subsystem from step 15 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
4. Complete the steps in Configuring the Developer Portal subsystem up to and including step 16 to configure the standard (non-2DCDR) settings for the portal subsystem in Dallas (DC1).
5. Configure the 2DCDR values to set Dallas to be active for the portal subsystem in DC1.
   Run the following commands:

   ```
   apicup subsys set port_dallas multi-site-ha-enabled=true

   apicup subsys set port_dallas multi-site-ha-mode=active

   apicup certs set port_dallas portal-replication-ingress --clear

   apicup subsys set port_dallas portal-replication=ptlreplicationdallas.cluster1.example.com

   apicup subsys set port_dallas replication-peer-fqdn=ptlreplicationraleigh.cluster2.example.com

   apicup subsys set port_dallas site-name=dallas
   ```

   where
   - ***port_dallas*** is the name of the portal subsystem that you are configuring.
   - ***ptlreplicationdallas.cluster1.example.com*** is the ingress hostname for the current data center in the two data center deployment.
   - ***ptlreplicationraleigh.cluster2.example.com*** is the ingress hostname for the other data center in the two data center deployment.
   - ***dallas*** is the name of the site in the current data center.

6. Complete the remainder of the steps in Configuring the Developer Portal subsystem from step 19 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
7. Check the status of the installation by running:

   ```
   apicup subsys status subsystem_name
   ```

8. Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

   Management subsystem endpoints
   - **platform-api**
   - **consumer-api**
   - **cloud-admin-ui**
   - **api-manager-ui**

   Portal subsystem endpoints
   - **portal-admin**
   - **portal-www**

- **Installing on the warm-standby data center**
  The following example shows how to set the 2DCDR values for the remote data center called **raleigh** (DC2), which is being installed as the warm-standby location.

  Important: Remember to use the same APICUP project directory as for the active data center. If you cannot use the same project directory because its location does not have network access to both data centers, you can copy the project directory over. Ensure that backups are kept of both project directories in each data center.

  1. Complete the steps in Configuring the Management subsystem up to and including step 14 to configure the standard (non-2DCDR) settings for the management subsystem in Raleigh (DC2).
  2. Configure the 2DCDR values to set the management subsystem to be the warm-standby in the Raleigh (DC2) data center.
     Run the following commands:

     ```
     apicup subsys set mgmt_raleigh multi-site-ha-enabled=true

     apicup subsys set mgmt_raleigh multi-site-ha-mode=passive

     apicup certs set mgmt_raleigh management-replication-ingress --clear

     apicup subsys set mgmt_raleigh management-replication=mgrreplicationraleigh.cluster2.example.com

     apicup subsys set mgmt_raleigh replication-peer-fqdn=mgrreplicationdallas.cluster1.example.com

     apicup subsys set mgmt_raleigh site-name=raleigh
     ```

     where
     - ***mgmt_raleigh*** is the name of the management subsystem that you are configuring.
     - ***mgrreplicationraleigh.cluster2.example.com*** is the ingress hostname for the current data center in the two data center deployment.
     - ***mgrreplicationdallas.cluster1.example.com*** is the ingress hostname for the other data center in the two data center deployment.
     - ***raleigh*** is the name of the site in the current data center.

  3. Set the encryption key manually for Raleigh (DC2) by copying it from Dallas (DC1).
     Run the following commands:

     ```
     apicup certs get mgmt_dallas encryption-secret -t key > mgmt-encryption-secret

     apicup certs set mgmt_raleigh encryption-secret mgmt-encryption-secret
     ```

  4. Check that the management encryption key is set the same on both DC1 and DC2.
     Run the following commands, and verify that the outputs match:

     ```
     apicup certs get mgmt_dallas encryption-secret -t key

     apicup certs get mgmt_raleigh encryption-secret -t key
     ```

  5. Complete the remainder of the steps in Configuring the Management subsystem from step 15 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.

6. Complete the steps in Configuring the Developer Portal subsystem up to and including step 16 to configure the standard (non-2DCDR) settings for the portal subsystem in Raleigh (DC2).
7. Configure the 2DCDR values to set Raleigh to be warm-standby for the portal subsystem on DC2.
   Run the following commands:

   ```
   apicup subsys set port_raleigh multi-site-ha-enabled=true

   apicup subsys set port_raleigh multi-site-ha-mode=passive

   apicup certs set port_raleigh portal-replication-ingress --clear

   apicup subsys set port_raleigh portal-replication=ptlreplicationraleigh.cluster2.example.com

   apicup subsys set port_raleigh replication-peer-fqdn=ptlreplicationdallas.cluster1.example.com

   apicup subsys set port_raleigh site-name=raleigh
   ```

   where
   - *port_raleigh* is the name of the portal subsystem that you are configuring.
   - *ptlreplicationraleigh.cluster2.example.com* is the ingress hostname for the current data center in the two data center deployment.
   - *ptlreplicationdallas.cluster1.example.com* is the ingress hostname for the other data center in the two data center deployment.
   - *raleigh* is the name of the site in the current data center.
8. Copy the encryption key for the portal subsystem on DC1 over to the portal subsystem on DC2.
   Run the following command to get the encryption key for the portal subsystem on DC1, and put it into a file that is called port-encryption-secret:

   ```
   apicup certs get port_dallas encryption-secret -t key > port-encryption-secret
   ```

   Run the following command to use the port-encryption-secret file to set the same encryption key on DC2:

   ```
   apicup certs set port_raleigh encryption-secret port-encryption-secret
   ```

   Where *port_dallas* is the name of the portal subsystem on DC1, and *port_raleigh* is the name of the portal subsystem on DC2.
   To check that the encryption key is set the same on both DC1 and DC2, you can run the following commands and verify that the outputs match:

   ```
   apicup certs get port_dallas encryption-secret -t key

   apicup certs get port_raleigh encryption-secret -t key
   ```

9. Complete the remainder of the steps in Configuring the Developer Portal subsystem from step 19 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
10. Check the status of the installation by running:

    ```
    apicup subsys status subsystem_name
    ```

- **Converting a single data center deployment to two data centers**
  Important:
  - Use the existing deployment as the active site. The new site must be set to warm-standby. If you set your existing deployment to warm-standby then all your management and portal data is erased.
  - The ingress endpoint values from the existing deployment become the global ingress endpoints for the two center deployment. The following endpoints must be set the same on your new deployment, as they are on your existing deployment:

    Management subsystem endpoints
    - `platform-api`
    - `consumer-api`
    - `cloud-admin-ui`
    - `api-manager-ui`

    Portal subsystem endpoints
    - `portal-admin`
    - `portal-www`

  - Configure your network to be able to route the endpoints to either data center, depending on which is the active.
  - You cannot configure a `site-name` for the existing management or portal cluster because the `site-name` property can be configured only at first deployment.

  The following example shows how to update the 2DCDR values to make the existing data center `Dallas` the active, and for installing and setting the new remote data center `Raleigh` as warm-standby.

  1. Configure the 2DCDR values to set Dallas to be active for the API Manager subsystem on DC1.
     Run the following commands:

     ```
     apicup subsys set mgmt_dallas multi-site-ha-enabled=true

     apicup subsys set mgmt_dallas multi-site-ha-mode=active

     apicup certs set mgmt_dallas management-replication-ingress --clear

     apicup subsys set mgmt_dallas management-replication=mgrreplicationdallas.cluster1.example.com

     apicup subsys set mgmt_dallas replication-peer-fqdn=mgrreplicationraleigh.cluster2.example.com
     ```

     where
     - *mgmt_dallas* is the name of the management subsystem that you are configuring.
     - *mgrreplicationdallas.cluster1.example.com* is the ingress hostname for the current data center in the two data center deployment.
     - *mgrreplicationraleigh.cluster2.example.com* is the ingress hostname for the other data center in the two data center deployment.

     Note: As the API Manager subsystem already exists, you cannot set the `site-name` property.
  2. Configure the 2DCDR values to set Dallas to be active for the portal subsystem on DC1.
     Run the following commands:

     ```
     apicup subsys set port_dallas multi-site-ha-enabled=true

     apicup subsys set port_dallas multi-site-ha-mode=active
     ```

```
apicup certs set port_dallas portal-replication-ingress --clear

apicup subsys set port_dallas portal-replication=ptlreplicationdallas.cluster1.example.com

apicup subsys set port_dallas replication-peer-fqdn=ptlreplicationraleigh.cluster2.example.com
```

where

- `port_dallas` is the name of the portal subsystem that you are configuring.
- `ptlreplicationdallas.cluster1.example.com` is the ingress hostname for the current data center in the two data center deployment.
- `ptlreplicationraleigh.cluster2.example.com` is the ingress hostname for the other data center in the two data center deployment.

Note: As the portal subsystem already exists, you cannot set the `site-name` property.

3. Run `apicup` to update the settings in each subsystem, for example:

```
apicup subsys install subsystem_name
```

where `subsystem_name` is the name of the management or portal subsystem that you want to update, for example `mgmt_dallas`.

4. Check the status of the installation by running:

```
apicup subsys status subsystem_name
```

5. Add the new DC, Raleigh (DC2) by completing the steps in Configuring the Management subsystem up to and including step 14 to create a management subsystem in Raleigh (DC2).
   Note:
   - Remember to use the same APICUP project directory as for the active data center.
   - Remember to set the same endpoints as for DC1:

     Management subsystem endpoints
     - `platform-api`
     - `consumer-api`
     - `cloud-admin-ui`
     - `api-manager-ui`

     Portal subsystem endpoints
     - `portal-admin`
     - `portal-www`

6. Configure the 2DCDR values to set Raleigh to be warm-standby for the API Manager subsystem on DC2.
   Run the following commands:

```
apicup subsys set mgmt_raleigh multi-site-ha-enabled=true

apicup subsys set mgmt_raleigh multi-site-ha-mode=passive

apicup certs set mgmt_raleigh management-replication-ingress --clear

apicup subsys set mgmt_raleigh management-replication=mgrreplicationraleigh.cluster2.example.com

apicup subsys set mgmt_raleigh replication-peer-fqdn=mgrreplicationdallas.cluster1.example.com

apicup subsys set mgmt_raleigh site-name=raleigh
```

   where
   - `mgmt_raleigh` is the name of the management subsystem that you are configuring.
   - `mgrreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.
   - `mgrreplicationdallas.cluster1.example.com` is the ingress hostname for the other data center in the two data center deployment.
   - `raleigh` is the name of the site in the current data center.

   Note: As this is a new deployment of the API Manager subsystem, you can set the `site-name` property.

7. Set the encryption key manually for Raleigh (DC2) by copying it from Dallas (DC1).
   Run the following commands:

```
apicup certs get mgmt_dallas encryption-secret -t key > mgmt-encryption-secret

apicup certs set mgmt_raleigh encryption-secret mgmt-encryption-secret
```

8. Check that the management encryption key is set the same on both DC1 and DC2.
   Run the following commands, and verify that the outputs match:

```
apicup certs get mgmt_dallas encryption-secret -t key

apicup certs get mgmt_raleigh encryption-secret -t key
```

9. Complete the remainder of the steps in Configuring the Management subsystem from step 15 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
10. Complete the steps in Configuring the Developer Portal subsystem up to and including step 16 to create a portal subsystem in Raleigh (DC2).
11. Configure the 2DCDR values to set Raleigh to be warm-standby for the portal subsystem on DC2.
    Run the following commands:

```
apicup subsys set port_raleigh multi-site-ha-enabled=true

apicup subsys set port_raleigh multi-site-ha-mode=passive

apicup certs set port_raleigh portal-replication-ingress --clear

apicup subsys set port_raleigh portal-replication=ptlreplicationraleigh.cluster2.example.com

apicup subsys set port_raleigh replication-peer-fqdn=ptlreplicationdallas.cluster1.example.com

apicup subsys set port_raleigh site-name=raleigh
```

    where
    - `port_raleigh` is the name of the portal subsystem that you are configuring.
    - `ptlreplicationraleigh.cluster2.example.com` is the ingress hostname for the current data center in the two data center deployment.

- ***ptlreplicationdallas.cluster1.example.com*** is the ingress hostname for the other data center in the two data center deployment.
- ***raleigh*** is the name of the site in the current data center.

12. Run the following command to generate the default portal certificates on the warm-standby site:

    ```
    apicup certs generate port_raleigh
    ```

13. Copy the encryption key for the portal subsystem on DC1 over to the portal subsystem on DC2.
    Run the following command to get the encryption key for the portal subsystem on DC1, and put it into a file that is called port-encryption-secret:

    ```
    apicup certs get port_dallas encryption-secret -t key > port-encryption-secret
    ```

    Run the following command to use the port-encryption-secret file to set the same encryption key on DC2:

    ```
    apicup certs set port_raleigh encryption-secret port-encryption-secret
    ```

    Where ***port_dallas*** is the name of the portal subsystem on DC1, and ***port_raleigh*** is the name of the portal subsystem on DC2.
    To check that the encryption key is set the same on both DC1 and DC2, you can run the following commands and verify that the outputs match:

    ```
    apicup certs get port_dallas encryption-secret -t key
    ```

    ```
    apicup certs get port_raleigh encryption-secret -t key
    ```

14. Complete the remainder of the steps in Configuring the Developer Portal subsystem from step 19 to verify that the configuration settings are valid, create your ISO file, and deploy the OVF template.
15. Check the status of the installation by running:

    ```
    apicup subsys status subsystem_name
    ```

16. Set your dynamic router to direct all traffic to DC1. The endpoints that must be directed to DC1 are:

Management subsystem endpoints
- `platform-api`
- `consumer-api`
- `cloud-admin-ui`
- `api-manager-ui`

Portal subsystem endpoints
- `portal-admin`
- `portal-www`

- **Validating your two data center deployment**
  Use the `apicup health-check` command to confirm that the management subsystems are ready and synchronized:

  ```
  apicup subsys health-check <management subsystem>
  ```

  If the management subsystem is healthy, the `health-check` command returns silently with no output.
  If you want to see more information in the `health-check` output, add the `-v` flag:

  ```
  apicup subsys health-check <management subsystem> -v
  ```

  ```
  ok(true) - Expected apicup version(10.0.7.0) to match apic version(10.0.7.0)
  ...
  ok(true) - ManagementCluster (specified multi site ha mode: active, current ha mode: active, ha status: Ready, ha message:
  Active DB is ready) is Ready | State: 17/17 | Phase: Running | Message: All services ready. HA status Ready - see HAStatus
  in CR for details
  ```

Example

Example of the management subsystem values that you can view by running the `apicup subsys get mgmt_name` command:

```
apicup subsys get mgmt_dallas
```

```
Appliance settings
==================

Name                          Value
Description
----                          -----
------
additional-cloud-init-file
(Optional) Path to additional cloud-init yml file
data-device                   sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld518TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/    (Optional)
Console login password for `apicadm` user, password must be pre-hashed
dns-servers                   [8.8.8.8]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network               172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network           172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                  eth0
Device for API/UI traffic (Eg: eth0)
search-domain                 [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                  [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                 eth0
Device for cluster traffic (Eg: eth0)
```

```
Subsystem settings
==================

Name                            Value
Description
----                            -----
------
deployment-profile              n1xc4.m16
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for portal
license-use                     production
License use (production/nonproduction)
multi-site-ha-enabled           true
Multi site HA enabled
multi-site-ha-mode              active
Multi site HA mode (active/passive)
replication-peer-fqdn           mgrreplicationraleigh.cluster2.example.com
Replication peer fully qualified name (replication endpoint of active mode site)
site-name                       dallas
Site name, used in k8s resource names
test-and-monitor-enabled        false
Test and Monitor enabled


Endpoints
=========

Name                            Value
Description
----                            -----
------
api-manager-ui                  api-manager-ui.testsrv0231.subnet1.example.com
FQDN of API manager UI endpoint
cloud-admin-ui                  cloud-admin-ui.testsrv0231.subnet1.example.com
FQDN of Cloud admin endpoint
consumer-api                    consumer-api.testsrv0231.subnet1.example.com
FQDN of consumer API endpoint
hub
FQDN of Test and Monitor hub endpoint, only required if Test and Monitor is enabled
management-replication          mgrreplicationdallas.cluster1.example.com
FQDN of Management replication endpoint, only required if HA is enabled
platform-api                    platform-api.testsrv0231.subnet1.example.com
FQDN of platform API endpoint
turnstile
FQDN of Test and Monitor turnstile endpoint, only required if Test and Monitor is enabled
```

Example of the portal subsystem values that you can view by running the `apicup subsys get` *`port_name`* command:

```
apicup subsys get port_dallas

Appliance settings
==================

Name                            Value
Description
----                            -----
------
additional-cloud-init-file
(Optional) Path to additional cloud-init yml file
data-device                     sdb
VM disk device (usually `sdb` for SCSI or `vdb` for VirtIO)
default-password
$6$rounds=4096$vtcqpAVK$dzqrOeYP33WTvTug38Q4Rld518TmdQgezzTnkX/PFwkzTiZ2S0CqNRr1S4b08tOc4p.OEg4BtzBe/r8RAk.gW/   (Optional)
Console login password for `apicadm` user, password must be pre-hashed
dns-servers                     [192.168.1.201]
List of DNS servers
extra-values-file
(Optional) Path to additional configuration yml file
k8s-pod-network                 172.16.0.0/16
(Optional) CIDR for pods within the appliance
k8s-service-network             172.17.0.0/16
(Optional) CIDR for services within the appliance
public-iface                    eth0
Device for API/UI traffic (Eg: eth0)
search-domain                   [subnet1.example.com]
List for DNS search domains
ssh-keyfiles                    [/home/vsphere/.ssh/id_rsa.pub]
List of SSH public keys files
traffic-iface                   eth0
Device for cluster traffic (Eg: eth0)


Subsystem settings
==================

Name                            Value
Description
----                            -----
------
deployment-profile              n1xc2.m8
Deployment profile (n1xc2.m16/n3xc4.m16) for analytics, (n1xc4.m16/n3xc4.m16) for management, (n1xc2.m8/n3xc4.m8) for portal
license-use                     production
```

```
License use (production/nonproduction)
multi-site-ha-enabled         true
Multi site HA enabled
multi-site-ha-mode            active
Multi site HA mode (active/passive)
replication-peer-fqdn         ptlreplicationraleigh.cluster2.example.com
Replication peer fully qualified name (replication endpoint of active mode site)
site-name                     dallas
Site name, used in k8s resource names


Endpoints
=========

Name                          Value
Description
----                          -----
------
portal-admin                  portal-api.example.com
FQDN of Portal admin endpoint
portal-replication            ptlreplicationdallas.cluster1.example.com
FQDN of Portal replication endpoint, only required if HA is enabled
portal-www                    portal-www.example.com
FQDN of Portal web endpoint
```

Related concepts

- Maintaining a two data center deployment

Related tasks

- How to failover a two data center deployment V10.0.6.0
- Recovering from a failover of a two data center deployment V10.0.6.0

Upgrading API Connect

Upgrade API Connect on VMware to a newer version.

Upgrading your API Connect deployment consists of three phases:

1. Upgrade planning. Decide on your upgrade path based on your current version.

   Review any feature changes in the new API Connect version that might impact how you use API Connect and plan for such changes.

   Upgrade planning should be started as early as possible to identify any issues that might influence when you are able to upgrade, and any actions to take before upgrade.

2. Upgrade preparation. Run pre-upgrade checks on your current deployment, download the new version of API Connect from Fix Central, backup your current deployment.

   Upgrade preparation can be done at any time, and it is worth doing early to identify and fix any potential problems before your upgrade window. If you do complete the upgrade preparation early, repeat the pre-upgrade checks just before your upgrade window to check for any new problems.

3. Upgrade process. Updating your API Connect subsystems to the new version.

   The upgrade process should be started only when all upgrade planning and preparation are completed. Interrupting and rolling back API Connect upgrades is not possible. Disaster recovery procedures must be followed to restore API Connect to the old version if problems are encountered during upgrade.

Use the procedures in this section to upgrade API Connect on VMware.

Note:

- For upgrading from v5, see Migrating a version 5 deployment to version 10.0.x.
- For information about how to maintain a two data center deployment, including upgrade considerations, see Maintaining a two data center deployment.

Important: API Connect v10.0.7.0 supports only backups to an S3 object-store for the management database. If your v10.0.6 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.
Upgrade to v10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store.

If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block. Support for SFTP and local backups will be provided in a future v10.0.x release.

- **Planning your API Connect upgrade on VMware**
  Before you upgrade API Connect on VMware, check that your deployment meets all the upgrade requirements.
- **Pre-upgrade preparation and checks VMware**
  Prepare your VMware deployment for upgrading to the newest version of API Connect.
- **Upgrading management, portal, and analytics on VMware**
  Upgrade your management, portal, and analytics subsystems on VMware.
- **Upgrading DataPower Gateway**
  Use these instructions to upgrade your DataPower Gateway services in a non-Kubernetes environment.
- **Troubleshooting upgrades on VMware**
  Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on VMware.

Planning your API Connect upgrade on VMware

Before you upgrade API Connect on VMware, check that your deployment meets all the upgrade requirements.

Supported upgrade paths

Table 1. Supported upgrade paths

Upgrade from:	How to upgrade to 10.0.7.0
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	Complete the procedure in Upgrading to the latest release on VMware.
10.0.5.2 or earlier	Complete two upgrades: 1. Upgrade to version 10.0.5.3 or later as explained in Upgrading to the latest release on VMware in the API Connect 10.0.5.x documentation. 2. Upgrade to the current release by completing the procedures in Upgrading to the latest release on VMware.

Supported DataPower Gateway versions

You can use any combination of API Connect CD releases (10.0.7.x) with DataPower CD releases (DataPower Gateway 10.5.3.x or DataPower API Gateway 10.5.3.x).

However, before you install, it is recommended to review the compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

Control planes for upgrading to 10.0.7

Installs with **`appliance-control-plane-1.27.x.tgz`**

Table 2. Control Plane files needed for upgrading to 10.0.7

Upgrading from API Connect version	Control Plane files needed
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	No Control Plane needed.

Upgrade requirements

- API Connect is not supported on a FIPS-enabled environment.
- The instructions for upgrading apply to the latest version of API Connect.
- The upgrade order for subsystems is important. Upgrade the subsystems in the following order:
  1. Management
  2. Portal
  3. Analytics
  4. Gateway

  Gateway must be upgraded after management so that any new policies and capabilities are available to any previously registered gateway services.

- Do not use **`kubectl exec`** commands to access API Connect pods unless advised by IBM.
  Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

Important upgrade considerations

Carefully review the following items, to plan and prepare for your upgrade.

OAuth provider resources with **`api.securityDefintions`**

If you have native OAuth providers that are configured with the **`api.securityDefintions`** field assigned, then upgrade fails. Before upgrade, remove all **`api.securityDefintions`** from all native OAuth providers that you configured in the Cloud Manager and API Manager UIs:

- Configuring a native OAuth provider in the Cloud Manager UI
- Configuring a native OAuth provider in the API Manager UI

Switch to the source view and delete any **`securityDefintions`** sections that are present.

SFTP management backups (V10.0.6.0 to V10.0.7.0 upgrade)

API Connect V10.0.7.0 supports only backups to an S3 object-store for the management database. If your V10.0.6.0 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.

Upgrade to V10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store.

If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block. Support for SFTP and local backups will be provided in a future V10.0.x release.

Extra backup folder appended to management database backup path

When you upgrade to V10.0.7.0 or later with S3 backups configured, an extra backup path folder /edb is appended to your backup path. The /edb path is to distinguish new EDB backups from your previous management backups.

Pre-upgrade preparation and checks VMware

Prepare your VMware deployment for upgrading to the newest version of API Connect.

Before you begin

Review the Planning your API Connect upgrade on VMware to ensure that you are following a supported upgrade path and that you understand important changes that might affect your deployment.

Procedure

VM health checks:

1. From your project directory, run the **apicup** health-check command for each of your subsystems.

   ```
   apicup subsys health-check <subsystem name>
   ```

   If the health check is successful, then the command returns silently.

   If you see an IO timeout error, then it's likely that a firewall is blocking access to port 9178. For example:

   ```
   apicup subsys health-check <subsystem name>
   ...
   Error: failed to install the subsystem: unable to open file send stream:
    rpc error: code = Unavailable desc = all SubConns are in TransientFailure,
    latest connection error: connection error:
    desc = "transport: Error while dialing dial tcp 1.2.3.4:9178: i/o timeout"
   ```

   If you see this error, make sure that your firewall has port 9178 open.

2. If you are upgrading to V10.0.7.0, and you have management database backups set to SFTP or local: Change your management database backup settings to S3.
   See Reconfiguring backup settings for the management subsystem V10.0.6.0.
   V10.0.7.0 supports only backups to an S3 object-store. If you are not able to configure an S3 object-store, then to upgrade to v10.0.7.0 you must enable the **accept-no-backups** setting before you upgrade, see step 4.a on Upgrading management, portal, and analytics on VMware.

3. Login to each of your API Connect VMs and switch to the root user:

   ```
   ssh apicadm@<hostname>
   ```

   ```
   sudo -i
   ```

4. Delete any stale upgrade files that might be left on the VM from previous upgrades:

   ```
   apic clean-upgrade-files
   ```

5. Remove obsolete **sources** files to ensure a clean upgrade.
   Delete any files in the directory /etc/apt/sources.list.d whose names match the format: repo-*<subsystem>*-*<version>*-*<appliance version>*.list:

   ```
   rm /etc/apt/sources.list.d/repo-*.list
   ```

6. Verify that each VM has sufficient disk space:
   a. Check the available space on the data disk:

      ```
      df -h /data/secure
      ```

      If disk usage is higher than 70%, then add disk capacity. See Adding disk space to a VMware appliance.

   b. Check available space in "/":

      ```
      df -h /
      ```

      If disk usage is higher than 70%, delete or copy off older **/var/log/syslog*.gz** files. The **syslog** files contain your API Connect and system logs. Do not delete any syslog files that might contain log data that is needed for any active support cases. If in doubt, backup the syslog files to another system before your delete them.

7. Run the **apicops** pre-upgrade health checks:
   Attention: Running the **apicops** pre-upgrade check is a required step. Failure to run this check before you start the upgrade might result in problems during the upgrade.
   a. Download the latest apicops release and copy it to your VM.
   b. On all subsystem VMs, as root user, run the subsystem status check:

      ```
      apicops upgrade:check-subsystem-status
      ```

      If the subsystem status check is successful, the command returns:

      ```
      All subsystems are in healthy, 'Running' state!
      ```

      If either check returns any errors, do not proceed with the upgrade.

   c. Run the **apicops** pre-upgrade check on your management VMs:
      Note: If you have a 2DCDR deployment, run this check only on the active management subsystem.

`apicops version:pre-upgrade`

If the `apicops` check returns:

`The original primary PVC is not attached to any pods in this namespace`

then follow the steps in The original primary PVC is not attached to any pods in this namespace.

If the `apicops` check returns:

`Current postgres leader is not the original leader, see https://ibm.biz/BdPLWU for next steps`

then follow the steps in the linked page https://ibm.biz/BdPLWU.

8. **Upgrading from v10.0.6:** Back up the Postgres images on one of the management VMs by running the following commands:

```
postgres_operator=$(kubectl get pod -l app.kubernetes.io/name=postgres-operator -o name)

ctr --namespace k8s.io images pull --plain-http=true `kubectl get  $postgres_operator  -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'`
ctr --namespace k8s.io images export rmdata.tar `kubectl get  $postgres_operator  -o jsonpath='{.spec.containers[].env[?
(@.name=="RELATED_IMAGE_PGO_RMDATA")].value}'`
ctr --namespace k8s.io images export backrestrepo.tar `kubectl get  $postgres_operator  -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_PGO_BACKREST_REPO")].value}'`
ctr --namespace k8s.io images export pgbouncer.tar `kubectl get  $postgres_operator  -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_CRUNCHY_PGBOUNCER")].value}'`
ctr --namespace k8s.io images export postgres-ha.tar `kubectl get  $postgres_operator  -o
jsonpath='{.spec.containers[].env[?(@.name=="RELATED_IMAGE_CRUNCHY_POSTGRES_HA")].value}'`

postgres_pod=$(kubectl get pod -l role=master -o name)
ctr --namespace k8s.io images export k8s-init.tar `kubectl get $postgres_pod  -o
jsonpath='{.spec.initContainers[].image}'`
```

Pre-upgrade backups and preparation:

9. Optional: Setup remote logging: Configuring remote logging for a VMware deployment.
   If the upgrade process fails, it is possible that important log messages for problem diagnosis are not retained on the VMs. Configure remote logging to ensure that all API Connect logging during the upgrade is retained.

10. Backup all your subsystems. Ensure that you have:
    - Database backups of all your subsystems.
    - Project directory backup.
    - **Optional:** VM snapshots of all subsystems.

    Upgrading from V10.0.7.0 or later: see Backing up and restoring on VMware.

    Upgrading from V10.0.6.0 or earlier: see Preparing for a disaster.

Download and verification of API Connect upgrade files:

11. Obtain the API Connect upgrade files:
    You can access the latest files from IBM Fix Central by searching for the API Connect product and your installed version.

    The following files are used during upgrade on VMware:

    IBM API Connect *<version>* Management Upgrade File for VMware
    : Management subsystem files for upgrade. Filename is: upgrade_management_*<version>*.

    IBM API Connect *<version>* Developer Portal Upgrade File for VMware
    : Developer Portal subsystem files for upgrade. Filename is: upgrade_portal_*<version>*

    IBM API Connect *<version>* Analytics Upgrade File for VMware
    : Analytics subsystem files for upgrade. Filename is upgrade_analytics_*<version>*

    IBM API Connect *<version>* Install Assist for *<operating_system_type>*
    : The apicup installation utility. Required for all installations on VMware. Filename is apicup-*<platform>*_*<version>*.

    The following files are not used during the upgrade process, but should be downloaded to replace your existing versions of them:

    IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
    : Toolkit command-line utility. Packaged as a stand-alone file, or with API Designer or Loopback:
      - IBM® API Connect *<version>* Toolkit for *<operating_system_type>*
      - IBM® API Connect *<version>* Toolkit with Loopback for *<operating_system_type>*
      - IBM® API Connect *<version>* Toolkit Designer with Loopback for *<operating_system_type>*

      Filename is toolkit-...*<operating system>*_*<version>*.
      Not required during initial installation. After installation, you can download directly from the Cloud Manager UI and API Manager UI. See Installing the toolkit.

    IBM® API Connect *<version>* Local Test Environment
    : Optional test environment. See Testing an API with the Local Test Environment
      Filename is apic-lte-*<operating system>*_*<version>*.

    IBM® API Connect *<version>* Security Signature Bundle File
    : Checksum files that you can use to verify the integrity of your downloads.
      Filename is signatures_*<version>*.

      To verify the files against the signatures, see Verifying the integrity of IBM product files.

12. If control plane files are required for your upgrade path, download them from the same IBM Fix Central page.
    Control plane files are required to support specific Kubernetes versions. The IBM API Connect *<version>* Management Upgrade File for VMware file contains the latest control plane file.

    Table 1 lists the control plane files that are needed for upgrading to the current release.

Tip: On the Fix Central page, type "control-plane" in the Filter fix details field to list only the control plane files in the results table. Then, click the Description header to sort the results so you can easily locate the control plane files that you need.

Table 1. Control plane files needed for upgrade

Version to upgrade from	Control plane files that are needed for upgrading to 10.0.7.0
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	No control plane files needed.

For information about the control plane files that are required for upgrading from older releases, see Control Plane files for earlier releases.

13. Verify that the subsystem upgrade files are not corrupted.
    a. Run the following command separately for the management, the developer portal, and the analytics upgrade files:
       - MacOS or Linux:

         ```
         sha256sum <upgrade-file-name>
         ```

         Example:

         ```
         sha256sum upgrade_management_v10.0.7.0
         ```

       - Windows:

         ```
         C:\> certUtil -hashfile C:\<upgrade-file-name> SHA256
         ```

         Example:

         ```
         C:\> certUtil -hashfile C:\upgrade_management_v10.0.7.0 SHA256
         ```

    b. Compare the result with the checksum values to verify that the files are not corrupted.
       If the checksum does not match the value for the corresponding version and subsystem, then the upgrade file is corrupted. Delete it and download a new copy of the upgrade file. The following list shows the checksum values for the current release.

       Checksum values for 10.0.7.0:

       ```
       10.0.7 analytics : 049f6c487ab62e07307c01206ce1ec75e2f2b1279ea1aa0f084d6220f5c206a4
       10.0.7 management : c8242110f1a5789ebe6a1dc1eefc17f0463451860794febb1f18969e7537dc0a
       10.0.7 portal : 09b23e0bc98ba2ba7b2ab4306bb4755acc95fb09bf5dc4fa8d09d453793dd04b
       ```

       For information about the checksum files used with older releases, see Checksum values for earlier releases.

14. Setup the new **apicup** command in your project directory.
    The **apicup** executable file is updated with each API Connect release. To upgrade API Connect, you must replace the **apicup** file from your current release with the new **apicup** file.
    Note: If you decide to stop or delay your API Connect upgrade, then revert these steps.
    a. In your project directory, take a backup of your current apicup file, appending the current API Connect version to the name of the backup file.
       For example:

       ```
       cp apicup apicup_v10.0.6.0
       ```

    b. Replace the apicup file in your project directory with the new apicup file that you downloaded in step 11.
    c. OSX and Linux® only: Make the apicup file executable:

       ```
       chmod +x apicup
       ```

    d. Set your operating system's `PATH` variable to include your project directory:

       Table 2. Commands for setting the path to the apicup file

Operating system	Command to set the path
macOS and Linux	`export PATH=$PATH:<project directory path>`
Windows	`set PATH=<project directory path>;%PATH%`

    e. Verify that the new **apicup** command is in your `PATH`, run:

       ```
       apicup version
       ```

       The output should show the version that you are upgrading to.

    f. Run the **apicup** command to accept the API Connect license:

       ```
       apicup licenses accept <License ID>
       ```

       <License ID> is specific to the API Connect Program Name that you purchased. To view the supported license IDs, see API Connect licenses.

- **Control Plane files for earlier releases**
  If you are upgrading to an earlier version than 10.0.7.0, refer to the following tables to determine which Control Plane files are required.
- **Checksum values for earlier releases**
  If you are upgrading to an earlier version than 10.0.7.0, use the corresponding checksum values to verify your downloaded upgrade files.

Control Plane files for earlier releases

If you are upgrading to an earlier version than 10.0.7.0, refer to the following tables to determine which Control Plane files are required.

Control planes for upgrading to 10.0.7

Installs with **appliance-control-plane-1.27.x.tgz**

Table 1. Control Plane files needed for upgrading to 10.0.7

Upgrading from API Connect version	Control Plane files needed
• 10.0.6.0 • 10.0.5.5 • 10.0.5.4 • 10.0.5.3	No Control Plane needed.

Control planes for upgrading to 10.0.6

Installs with **appliance-control-plane-1.26.x.tgz**

Table 2. Control Plane files needed for upgrading to 10.0.6

Upgrading from API Connect version	Control Plane files needed
• 10.0.5.3	No Control Plane needed.
• 10.0.5.2	Download: • **appliance-control-plane-1.25.x.tgz**
• 10.0.5.1	Download: • **appliance-control-plane-1.25.x.tgz** • **appliance-control-plane-1.24.x.tgz**

Checksum values for earlier releases

If you are upgrading to an earlier version than 10.0.7.0, use the corresponding checksum values to verify your downloaded upgrade files.

```
10.0.7 analytics : 049f6c487ab62e07307c01206ce1ec75e2f2b1279ea1aa0f084d6220f5c206a4
10.0.7 management : c8242110f1a5789ebe6a1dc1eefc17f0463451860794febb1f18969e7537dc0a
10.0.7 portal : 09b23e0bc98ba2ba7b2ab4306bb4755acc95fb09bf5dc4fa8d09d453793dd04b

10.0.6 analytics : d927781a01ed2172ae978f83bc17e8b7e12994f8de6027fab2124adf2d40f09c
10.0.6 management : 876ca328aa1e411d8e3896b65e3dff925baaf85579b1ac65248749b321126436
10.0.6 portal : a2bf468ce1bc7c6104858d757088364001e8620922396b80b52657d146c303de
```

Upgrading management, portal, and analytics on VMware

Upgrade your management, portal, and analytics subsystems on VMware.

Before you begin

- Review Planning your API Connect upgrade on VMware.
- Complete the Pre-upgrade preparation.
- Enable keep-alive in your SSH client configuration to avoid disconnects during the transfer of the upgrade images to your VMs. For more information, consult the documentation for the version of SSH that is installed on your client system (where you run the **apicup** commands from).

About this task

- If you have a two data center disaster recovery deployment, the upgrade steps are slightly different. Refer to Upgrading a two data center deployment.
- The upgrade order for subsystems is important. Upgrade the subsystems in the following order: (1) management, (2) portal, (3) analytics, (4) gateway. Management must be upgraded first to ensure that any new policies and capabilities are available to previously registered gateway services.
- When you start the upgrade process on a subsystem, the **apicup install** command sends the compressed upgrade files to all subsystem VMs. The compressed files are about 2 GB, and the transfer can take some time. The **apicup install** command exits when the transfer of the upgrade files to each VM is complete, at which point the upgrade starts. The progress of the upgrade is monitored with other commands.
- The **apicup subsys install** command runs **apicup health-check** first before the upgrade process is started. If the health-check fails, then the upgrade is aborted and details of the health-check failure is displayed.
  If you are troubleshooting a problem with upgrade and want the upgrade to proceed despite a health-check failure, then you can suppress the health check. See Skipping health check when re-running upgrade.
- You do not need to upgrade the components API Connect Toolkit and API Connect Local Test Environment in the same upgrade window. These components are single executable files. Replace them the new versions downloaded from Fix Central.

Procedure

Notes:

- If a backup of any of your subsystems is scheduled to run within a few hours, do not start an upgrade.
- While the upgrade process is running, do not do any maintenance tasks such as rotating key-certificates, or restoring from a backup.

- If you encounter an issue during the upgrade, see Troubleshooting upgrades on VMware.

1. If some time has passed since you completed the Pre-upgrade preparation, repeat these pre-upgrade steps to be sure that your subsystems are ready for upgrade.
2. Block all external traffic to the management server.
   Upgrading API Connect requires downtime while files are transferred and updated. All external traffic to the management server from all sources (Cloud Manager UI, API Manager UI, CLI, and REST API) must be blocked. As a result, the Cloud Manager and API Manager UIs cannot be used during the upgrade. In addition, the developer portal cannot be used to create, update, or delete any applications, subscriptions, memberships, or consumer organizations during the upgrade.

3. Verify that the new **apicup** command is in your operating system's `PATH` variable:

   ```
   apicup version
   ```

   The version that is returned should be the version that you are upgrading to. If not, then refer to step 14 on Pre-upgrade preparation and checks VMware.

4. Upgrade the management subsystem:
   a. If you are upgrading to v10.0.7.0 and are not able to configure S3 object-store backups for your management subsystem, then set `accept-no-backups` to `true` to allow upgrade to proceed:

      ```
      apicup subsys set <subsystem_name> accept-no-backups=true
      ```

      For more information about this limitation, see Planning your API Connect upgrade on VMware.
   b. Start the upgrade of the management subsystem:
      i. Install the management subsystem files.
      Note:
      If you have a two data center disaster recovery deployment and are upgrading to v10.0.7.0, all data on the warm-standby management subsystem is deleted during the upgrade, to be replaced by data from the upgraded active management subsystem.

      Include the flag `--accept-warm-standby-upgrade-data-deletion` in the `apicup subsys install <mgmt_subsystem> <upgrade_tar>` command to acknowledge this temporary data deletion.

      If you have a two data center disaster recovery deployment, and are upgrading to v10.0.6.0, do not follow this step. Instead, follow the steps in Management subsystem two data center upgrade from v10.0.5.

      For more information about two data center disaster recovery upgrade, see Upgrading a two data center deployment.

      If you are upgrading to v10.0.7.0, include the `--skip-health-check` argument:

      ```
      apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> --skip-health-check
      ```

      For all other upgrade paths, use:

      ```
      apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
      ```

      Notes:
      - If you are adding one or more control plane files, specify the file path to each control plane file:

        ```
        apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1> <path_to_control_plane_file_n>
        ```

      - If the upgrade gets stuck, run the following command to restart the appliance:

        ```
        sudo systemctl restart appliance-manager
        ```

      ii. Verify that the upgrade was successful by running the **apicup** health check:

      ```
      apicup subsys health-check <subsystem_name>
      ```

      iii. If upgrading to 10.0.7.0, verify that the health check did not return a false-positive response.
      Note: This step only applies if you upgraded to API Connect 10.0.7.0 and is due to a known issue that will be corrected in a later release.
      Run the following command to get the current version of API Connect on the upgraded subsystem:
      1. Log in to the VM:

         ```
         ssh apicadm@<hostname>
         ```

      2. Switch to `root` user:

         ```
         sudo -i
         ```

      3. Get the API Connect version:

         ```
         kubectl get apic
         ```

      If the health check indicates a successful upgrade but the subsystem is still running the older version of API Connect, skip the remaining steps in this procedure and see False positive result from health check after upgrading to 10.0.7.0 for troubleshooting instructions.

      If the subsystem is running the newer version of API Connect then the upgrade was successful; continue with the remaining steps in this procedure.

      iv. If the health check indicates that a reboot is needed, complete the following steps on all management VMs:
      1. Login to the VM:

         ```
         ssh apicadm@<hostname>
         ```

      2. Switch to `root` user:

         ```
         sudo -i
         ```

      3. `systemctl reboot`

   c. Use `apicops` to validate the certificates.
      For more information about the `apicops` tool, see The API Connect operations tool: apicops.

i. Login to the VM:

```
ssh apicadm@<hostname>
```

ii. Switch to `root` user:

```
sudo -i
```

iii. Run the following command:

```
apicops upgrade:stale-certs
```

iv. Delete any stale cert-manager certificates.
If a cert-manager certificate failed validation, delete the stale certificate secret to trigger cert-manager to regenerate it. Run the following command:

```
kubectl delete secret <stale-secret>
```

v. Restart the corresponding pod so that it can pick up the new secret.
To determine which pod to restart, see the following topics:
- List of ingress certificates
- List of intra-subsystem certificates

5. If you have multiple management subsystems in the same project, set the portal subsystem's `platform-api` and `consumer-api` certificates to match the certificates used by their corresponding management subsystems.
This step applies only if you installed more than one management subsystem into a single project.

   A portal subsystem can be associated with only one management subsystem. To associate the upgraded portal subsystem with the appropriate management subsystem, manually set the `mgmt-platform-api` and `mgmt-consumer-api` certificates to match the ones used by the management subsystem.

   a. Run the following commands to get the certificates from the management subsystem:

   ```
   apicup certs get mgmt-subsystem-name platform-api -t cert > platform-api.crt
   apicup certs get mgmt-subsystem-name platform-api -t key > platform-api.key
   apicup certs get mgmt-subsystem-name platform-api -t ca > platform-api-ca.crt

   apicup certs get mgmt-subsystem-name consumer-api -t cert > consumer-api.crt
   apicup certs get mgmt-subsystem-name consumer-api -t key > consumer-api.key
   apicup certs get mgmt-subsystem-name consumer-api -t ca > consumer-api-ca.crt
   ```

   where *`mgmt-subsystem-name`* is the name of the specific management subsystem that you want to associate the new portal subsystem with.

   b. Run the following commands to set the portal's certificates to match the certificates used by the management subsystem:

   ```
   apicup certs set ptl-subsystem-name platform-api Cert_file_path Key_file_path CA_file_path

   apicup certs set ptl-subsystem-name consumer-api Cert_file_path Key_file_path CA_file_path
   ```

6. Upgrade the portal subsystem.
Attention: If you are upgrading a two data center disaster recovery deployment, upgrade the warm-standby data center first.
For more information about two data center disaster recovery upgrade, see Upgrading a two data center deployment.

   a. Install the portal subsystem files:

   ```
   apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
   ```

   Note: If you are adding one or more control plane files, specify the file path of each control plane file:

   ```
   apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1>
   <path_to_control_plane_file_n>
   ```

   b. Verify that the upgrade was successful by running a health check with the following command:

   ```
   apicup subsys health-check <subsystem_name>
   ```

   Important: If the health check indicates that a reboot is required, do not reboot at this stage. Reboot after all sites are upgraded, see step 7.

   c. Verify that the health check did not return a false-positive response.
   Run the following command to get the current version of API Connect on the upgraded subsystem:
      i. Log in to the VM:

      ```
      ssh apicadm@<hostname>
      ```

      ii. Switch to `root` user:

      ```
      sudo -i
      ```

      iii. Get the API Connect version:

      ```
      kubectl get apic
      ```

   If the health check indicates a successful upgrade but the subsystem is still running the older version of API Connect, skip the remaining steps in this procedure and see False positive result from health check after upgrading to 10.0.7.0 for troubleshooting instructions.

   If the subsystem is running the newer version of API Connect then the upgrade was successful; continue with the remaining steps in this procedure.

   d. If you have two data center disaster recovery deployment and upgraded the portal subsystem in the warm-standby data center, then start the upgrade on the active data center.
   Note: In a two data center disaster recovery deployment, after the warm-standby is upgraded, the active data center might enter a non-READY state. If this happens, do not be concerned, proceed to upgrade the portal subsystem in the active data center.

7. Verify portal sites are upgraded.
Run the following toolkit commands to ensure that the portal site upgrades are complete:

   a. Log in as an admin user:

```
apic login -s <server_name> --realm admin/default-idp-1 --username admin --password <password>
```

b. Get the portal service ID and endpoint:

```
apic portal-services:get -o admin -s <management_server_endpoint> \
            --availability-zone availability-zone-default <portal-service-name> \
            --output - --format json
```

c. List the sites:

```
apic --mode portaladmin sites:list -s <management_server_endpoint> \
            --portal_service_name <portal-service-name> \
            --format json
```

Any sites currently upgrading display the **UPGRADING** status; any site that completed its upgrade displays the **INSTALLED** status and the new platform version. Verify that all sites display the **INSTALLED** status before proceeding.

For more information about the **sites** command, see **apic sites:list** and Using the **sites** commands.

d. After all sites are in **INSTALLED** state and show the new platform listed, run:

```
apic --mode portaladmin platforms:list -s <server_name> --portal_service_name <portal_service_name>
```

Verify that the new version of the platform is the only platform listed.

For more information about the **platforms** command, see **apic platforms:list** and Using the **platforms** commands.

e. If all portal sites are upgraded and the health check indicates that a reboot is needed, then run the following command on all portal VMs:

```
systemctl reboot
```

8. Upgrade the analytics subsystem.
   a. Install the analytics subsystem files:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive>
```

Note: If you are adding one or more control plane files, specify the file path of each control plane file:

```
apicup subsys install <subsystem_name> <subsystem_upgrade_tar_archive> <path_to_control_plane_file_1>
<path_to_control_plane_file_n>
```

b. Verify that the upgrade was successful by running a health check with the following command:

```
apicup subsys health-check <subsystem_name>
```

c. If upgrading to 10.0.7.0, verify that the health check did not return a false-positive response.
Note: This step only applies if you upgraded to API Connect 10.0.7.0 and is due to a known issue that will be corrected in a later release.
Run the following command to get the current version of API Connect on the upgraded subsystem:

```
kubectl get apic
```

If the health check indicates a successful upgrade but the subsystem is still running the older version of API Connect, skip the remaining steps in this procedure and see False positive result from health check after upgrading to 10.0.7.0 for troubleshooting instructions.

If the subsystem is running the newer version of API Connect then the upgrade was successful; continue with the remaining steps in this procedure.

d. If the health check indicates that a reboot is needed, run the following commands on all analytics VMs:

```
systemctl reboot
```

9. Take a backup of your project directory, and database backups of each upgraded API Connect subsystem; see Backing up and restoring on VMware.
10. Optional: Take a virtual machine (VM) snapshot of all your VMs; see Using VM snapshots for infrastructure backup and disaster recovery for details. This action requires a brief outage while the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully.

What to do next

When your management, portal, and analytics subsystem are upgraded, upgrade your DataPower Gateway Service. See Upgrading DataPower Gateway.

Upgrading DataPower Gateway

Use these instructions to upgrade your DataPower Gateway services in a non-Kubernetes environment.

Before you begin

Complete the upgrade of the API Connect management, portal, and analytics subsystems before upgrading gateway. Upgrading the management subsystem before the gateway ensures that any new policies and capabilities are available to a previously registered gateway service.

About this task

- The DataPower Gateway Service in a non-Kubernetes environment can be on either a physical appliance or in a virtual DataPower deployment.
- The upgrade of DataPower Gateway Service in a non-Kubernetes environment, for use in API Connect deployments in a VMware environment, does not use the **apicup** command. Use DataPower Gateway instructions in the following steps.

Note: To upgrade the DataPower Gateway Service in a Kubernetes environment, do not use this procedure. Instead, see Upgrading the gateway subsystem on Kubernetes.

Procedure

1. Ensure that the DataPower Gateway firmware version you plan to install is compatible with the API Connect management subsystem version.
   Note:
   You can use any combination of API Connect CD releases (10.0.6.x) with DataPower CD releases (DataPower Gateway 10.5.1.x or DataPower API Gateway 10.5.1.x).

   Before you install, best practice is to review the latest compatibility support for your version of API Connect. To view compatibility support, follow the instructions in IBM API Connect Version 10 software product compatibility requirements to access API Connect information on the Software Product Compatibility Reports website. Once you access information for your version of API Connect, select Supported Software > Integration Middleware, and view the list of compatible DataPower Gateway versions.

2. When you upgrade a high-availability cluster, check that you meet the following requirements:
   - Gateways must be updated one at a time.
   - Before you start the upgrade, a single gateway must be running as primary for all gateway-peering definitions.
   - If you are upgrading multiple gateways, the primary gateway must be upgraded last.

   To determine which gateway is running as primary, use either the **`show gateway-peering-status`** command in the DataPower CLI, or use the Gateway Peering Status display in the Web GUI in the API Connect application domain. To move the primary to the DataPower on which you are working, you can issue the **`gateway-peering-switch-primary <peering-object-name>`** command.

3. Ensure that your deployment includes an NTP server to synchronize time between each of the DataPower Gateways.
   See Managing the NTP service.
4. Ensure that a unique **`System Identifier`** is set for each v10 DataPower Gateway . See Initializing the DataPower Gateway.
5. Follow the upgrade instructions on the DataPower Gateway documentation. See Installation operations.
   Note:
   - If you are upgrading a cluster (high-availability) deployment, ensure that you have identified which gateway is running a primary in gateway-peering definitions, and that you upgrade that gateway last.
   - For troubleshooting help with DataPower operator deployments, see IBM DataPower Operator documentation.

Upgrading a two data center deployment

How to upgrade a two data center disaster recovery (DR) deployment on VMware.

To upgrade a two data center disaster recovery deployment, follow the API Connect on VMware upgrade procedures along with the additional procedures that are described in this topic.

Key points for two data center disaster recovery upgrade on VMware:

- Both data centers must be using the same version of API Connect, including down to the specific fix pack and interim fix level. Therefore, both data centers must be upgraded in the same maintenance window.
- Do not reboot any of the management or portal virtual machines until the end of the upgrade, even if you're instructed to do so by the **`apicup subsys health-check`** command during the upgrade process.
- The management and portal subsystems in the warm-standby data center must be upgraded first.
- Upgrades to V10.0.6.0 only: 2DCDR is disabled on the management subsystems during the upgrade process. Keep a note of which data center had the active management subsystems when you start the upgrade process; set this same data center as the active at the end of the process.

Upgrade steps for 2DCDR:

1. Review the Planning your API Connect upgrade on VMware.
2. Complete the Pre-upgrade preparation and checks VMware steps.
3. **Upgrades to v10.0.7 and later only:** Upgrade the warm-standby management subsystem first, include the **`--accept-warm-standby-upgrade-data-deletion`** flag in the upgrade command.
4. **Upgrades to v10.0.6 only:** To upgrade the management subsystem, follow Management subsystem two data center upgrade from v10.0.5.
5. To upgrade the portal subsystem, follow the steps in Upgrade on VMware, but ensure that the warm-standby is upgraded first.
6. Upgrade analytics and gateway subsystems as described in Upgrade on VMware.

How to upgrade when one data center is down

If you cannot get your failed data center back online and want to upgrade, you can convert your active data center to a stand-alone and upgrade it:

1. Ensure that the network links to the failed data center are removed, so that if the failed data center recovers unexpectedly, it cannot interfere with your active data center.
2. Convert the active data center to a stand-alone data center: Removing a two data center deployment.
3. Upgrade your active data center with the standard upgrade steps: Upgrade on VMware.

To restore 2DCDR, replace the API Connect deployment in your failed data center with a new installation at the same fix pack level as your upgraded active data center. Follow the steps in Installing a two data center deployment to configure 2DCDR between the data centers.

Related concepts

- Two data center warm-standby deployment strategy on VMware

Related tasks

- Installing a two data center deployment

Management subsystem two data center upgrade from v10.0.6

Upgrades from v10.0.6 and later: How to upgrade your management subsystem in a two data center deployment.

About this task

Follow the steps in this topic, along with the steps documented in Upgrading to the latest release on VMware.

Some steps in the upgrade process are run from the project directory with the **apicup** command. Other steps require you to login to the management VMs as root user. To login as root, follow these steps:

- Use SSH to login to the VM as the `apicadm` user:

  ```
  ssh apicadm@<hostname>
  ```

- Switch to the root user with `sudo`:

  ```
  sudo -i
  ```

Procedure

1. Review the Planning your API Connect upgrade on VMware.
2. Complete the Pre-upgrade preparation and checks VMware steps.
3. Block all external traffic to the management server.
   Upgrading API Connect requires downtime while files are transferred and updated. All external traffic to the management server from all sources (Cloud Manager UI, API Manager UI, CLI, and REST API) must be blocked. As a result, the Cloud Manager and API Manager UIs cannot be used during the upgrade. In addition, the developer portal cannot be used to create, update, or delete any applications, subscriptions, memberships, or consumer organizations during the upgrade.
4. Verify that the new **apicup** command is in your operating system's `PATH` variable:

   ```
   apicup version
   ```

   The version that is returned should be the version that you are upgrading to. If not, then refer to step 14 on Pre-upgrade preparation and checks VMware.
5. Upgrade your warm-standby first. Run the upgrade command with the `--accept-warm-standby-upgrade-data-deletion` and `--skip-health-check` flags:

   ```
   apicup subsys install <subsystem name> <subsystem upgrade tar archive> --accept-warm-standby-upgrade-data-deletion --skip-health-check
   ```

   Note: If any Control Plane files are needed for your upgrade, you must append the files to the `install` command. For example:

   ```
   apicup subsys install <subsystem name> <subsystem upgrade tar archive> <path to control plane file 1> <path to control plane file 2> --accept-warm-standby-upgrade-data-deletion --skip-health-check
   ```

   The `--accept-warm-standby-upgrade-data-deletion` flag is required to acknowledge that you accept that the management subsystem data is deleted from the warm-standby during the upgrade. The deleted data is replaced when the active management subsystem is upgraded and synchronizes with the upgraded warm-standby. W
6. Login to the warm-standby VM to monitor the status. Verify that the subsystem status reaches the `Running` state.

   ```
   kubectl get mgmt
   ```

   ```
   NAME          READY   STATUS    VERSION    RECONCILED VERSION AGE
   management    n/n     Running   <current version>   <current version>   5h2m
   ```

   The `RECONCILED VERSION`/`VERSION` should still show your current version, not the version you are upgrading to.
7.
8. Run the upgrade command on both sites without the `--skip-operand-update` flag:

   ```
   apicup subsys install <subsystem name> <subsystem_upgrade_tar_archive>
   ```

9. Continue following the standard management subsystem upgrade steps, from step 5.b.ii on Upgrade on VMware.

Management subsystem two data center upgrade from v10.0.5

Upgrades to v10.0.6 from v10.0.5 only: How to upgrade your management subsystem in a two data center deployment.

About this task

Follow the steps in this topic, along with the steps documented in Upgrading to the latest release on VMware.

The steps in this topic involve running command-line operations from several locations:

- **apicup** operations are run from your API Connect project directory.
  Note: The **apicup** command is a binary file that is included with each API Connect release. Unless stated otherwise, all operations in this topic that use the **apicup** command must use the **apicup** binary from the target release (the release that you want to upgrade to). Be careful to not use the **apicup** command from your

source release.
- **apic** and **kubectl** operations are run from within your management appliance, as the root user. Login to your management virtual appliance with an SSH client and the username `apicadm`, then switch to the root user with `sudo -i`.

Procedure

1. Determine the current version and license of API Connect (the version that you are upgrading from).
   You will need to specify the correct version number and license in the next step.
   a. To get the current version, from within the management appliance run the following command:

      ```
      apic version
      ```

      In the response, the `- subsystem-<subsytem-name>` statement displays the installed version. In the following example, the response indicates that the installed version is 10.0.5.3:

      ```
      apic version

      INFO[0000] Log level: info
      Control plane: 1.26.0
      Appliance base 4.6.0
      GitCommitSha:10.0.5.2-55-gb784ac7
      Packages:
      - appliance-base:4.6.0
      - appliance-runtime-4.6.0:1.0.0
      - appliance-setup-4.6.0:1.0.0
      - subsystem-management:1:10.0.5.3
      ```

   b. To get the current license, run the following **apicup** command in your project directory:

      ```
      apicup licenses list
      ```

      in the response, the `ID` statement displays the installed license, for example:

      ```
      Accepted licenses:
              ID: <source_license>
              Product Version: <source_version>
              URL: <license_url>
      ```

      Note: If there are multiple accepted licenses, refer only to the `<source_license>` that aligns with the `<source_version>` of your system (that is the `<source_version>` that matches the version that you obtained in step 1.a).
2. Add source license and version to your extra-values.yaml file.
   Note: If you do not have an extra-values.yaml file, then create one in a directory that the `apicup` command can access.
   Add the source lines that you identified in step 1 to your extra-values.yaml as shown:

   ```
   spec:
     version: "<source_version>"
     license:
       license: <source_license>
     imageRegistry: 127.0.0.1:8675/<source_version>
   ```

   <source_version> is added in two places: `version` and `imageRegistry`.
3. Apply the extra-values.yaml file.
   Run the following `apicup` command to set the `extra-values` file for both the `active` and `warm-standby` sites:

   ```
   apicup subsys set <mgmt> extra-values-file=<extra-values-file-path>
   ```

   where *<mgmt>* is the name of your management subsystem, which you can get with:

   ```
   apicup subsys list
   ```

   If you want to validate your `<extra-values-file-path>`, and the other subsystem settings, before proceeding, you can run:

   ```
   apicup subsys get <mgmt> --validate
   ```

4. Start the upgrade process on both active and warm-standby management subsystems.

   ```
   apicup subsys install <mgmt> <mgmt_upgrade_tar_archive>
   ```

   Note: If any Control Plane files are needed for your upgrade, you must append the files to the `install` command.
5. Wait for the management subsystem status to become `Running` on both sites.
   To determine the status, login to the management appliances on both sites as root user and run:

   ```
   kubectl get mgmt

   NAME         READY   STATUS    VERSION    RECONCILED VERSION AGE
   management   n/n     Running   10.0.6.0   10.0.6.0-100   5h2m
   ```

   The `RECONCILED VERSION`/`VERSION` should be the `source_version` that you retrieved in step 1.

6. Disable 2DCDR on the `warm-standby` management subsystem:

   ```
   apicup subsys set <warm-standby management subsystem> multi-site-ha-enabled=false
   apicup subsys install <warm-standby management subsystem>
   ```

   Conversion from warm-standby to a stand-alone management subsystem can take some time. Run the health check to ensure that the deployment is in a good state before proceeding:

   ```
   apicup subsys health-check <warm-standby management subsystem>
   ```

an empty response means no detected problems.

7. Disable 2DCDR on the active management subsystem.

```
apicup subsys set <active management subsystem> multi-site-ha-enabled=false
apicup subsys install <active management subsystem>
```

Conversion from active to a stand-alone management subsystem can take some time. Run the health check to ensure that the deployment is in a good state before proceeding:

```
apicup subsys health-check <active management subsystem>
```

an empty response means no detected problems.

8. Wait for the management subsystem status to become **Running** on both sites.
To determine the status, login to the management appliances on both sites as root user and run:

```
kubectl get mgmt
```

```
NAME          READY   STATUS    VERSION    RECONCILED VERSION AGE
management    n/n     Running   10.0.6.0   10.0.6.0-100    5h2m
```

The **RECONCILED VERSION**/**VERSION** should be the ***source_version*** that you retrieved in step 1.

9. Edit your extra-values.yaml file and delete the **version**, **license**, and **imageRegistry** properties that you added in step 2.
If you have no other properties in your extra-values.yaml file, then unset the **extra-values-file** property:

```
apicup subsys set mgmt extra-values-file=
```

Apply the change to both sites by running:

```
apicup subsys install <mgmt>
```

10. Wait for the management subsystems status to become **Running** on both sites.

```
kubectl get mgmt
NAME          READY   STATUS    VERSION    RECONCILED VERSION AGE
management    16/16   Running   10.0.6.1   10.0.6.1-100    5h2m
```

The **RECONCILED VERSION**/**VERSION** should now be the target release that you are upgrading to.

11. Run the health check on both clusters to ensure that the upgrade completed.

```
apicup subsys health-check <mgmt>
```

an empty response means no detected problems.

12. Enable 2DCDR on the previously **active** management subsystem.

```
apicup subsys set <active management subsystem> multi-site-ha-enabled=true
apicup subsys install <active management subsystem>
```

13. Wait for the management subsystem status to become **Running** on the active site.

```
kubectl get mgmt
NAME          READY   STATUS    VERSION    RECONCILED VERSION AGE
management    16/16   Running   10.0.6.1   10.0.6.1-100    5h2m
```

The **RECONCILED VERSION**/**VERSION** should now be the target release that you are upgrading to.

14. Enable 2DCDR on the previously warm-standby management subsystem.

```
apicup subsys set <warm-standby management subsystem> multi-site-ha-enabled=true
apicup subsys install <warm-standby management subsystem>
```

15. Wait for the management subsystem status to become **Running** on the warm-standby site.

```
kubectl get mgmt
NAME          READY   STATUS    VERSION    RECONCILED VERSION AGE
management    16/16   Running   10.0.6.1   10.0.6.1-100    5h2m
```

The **RECONCILED VERSION**/**VERSION** should now be the target release that you are upgrading to.

16. Verify that 2DCDR replication is working.
To validate that your two data center deployment is replicating data, login to your active management appliance as root user, run **kubectl get mgmt**:

```
kubectl get mgmt
```

```
NAME          READY   STATUS    VERSION      RECONCILED VERSION   MESSAGE
AGE
management    n/n     Running   10.0.5.3-0   10.0.5.3-0           Management is ready.              8m59s
```

If you see **HA Status Warning** in the **MESSAGE** column, then check the status details with **kubectl describe mgmt**:

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

Notes:

- It takes time for the **warm-standby** site to sync with the **active** site.
- When **STATUS** shows **Running**, you can reboot (if **apicup subsys health-check** advises you to do so).

Troubleshooting upgrades on VMware

Review the following troubleshooting guidance if you encounter a problem while upgrading API Connect on VMware.

For additional issues that you might encounter when upgrading to v10.0.6.0 only, see Troubleshooting upgrades to v10.0.6.0 on VMware.

Subsystem is stuck in `Pending` state with a reason of `PreUpgradeCheckInProgress`

Before the subsystem microservices are upgraded, the operator triggers a set of pre-upgrade checks that must pass for the upgrade to proceed. If one or more of the checks fail, the subsystem status remains in **Pending** state with a reason of **PreUpgradeCheckInProgress**. Check the status condition of the subsystem CR to confirm the pre-upgrade check failed. The **status.PreUpgradeCheck** property contains a summary of the failed checks. Full logs for the checks that are carried out can be viewed in the **ConfigMap** referenced in the **status.PreUpgradeCheck** property. The pre-upgrade checks automatically retry until they successfully pass. If you are unable to rectify the problem that causes a check to fail, then open an IBM support case.

False positive result from health check after upgrading to 10.0.7.0

Sometimes an upgrade to API Connect 10.0.7.0 appears to complete successfully and the health-check reports that the upgraded subsystem is healthy, but the older version is still installed. A false positive response from the health check indicates a problem with the upgrade.

Note: This step only applies if you upgraded to API Connect 10.0.7.0 and is due to a known issue that will be corrected in a later release.
Complete the following steps to resolve the issue:

1. If you did not do so already, run the follow command on each subsystem to determine what version of API Connect is deployed:

   ```
   kubectl get apic
   ```

   If the new version of API Connect is deployed on a subsystem, then the upgrade was successful for that subsystem.

2. If the older version of API Connect is still deployed on a subsystem, run the following command to check the status of that subsystem in case there is a message about the underlying problem:

   ```
   apic status
   ```

   Sometimes the status indicates the problem and you can correct it before upgrading that subsystem again.
3. If you do not see any messages indicating the source of the problem, contact IBM Support for assistance.

V10.0.7 only: Upgrade issues a warning for management subsystem to use S3 backups

If you previously used SFTP for backups, you must configure S3 backups before upgrading to v10.0.7.0. Configuring the S3 backups requires that you use the older version of the **apicup** tool (from your current release of API Connect) to both set the values and run the **apicup install** command.

If you used the wrong version of **apicup** to configure S3 backup settings before upgrading, you might see a warning for the Management subsystem with the following message: **Please set S3 backup configs.**

Correct the problem by running the following command run the upgrade again but skip the health-check and allow the CR to be updated with the new S3 settings:

```
./apicup subsys install <mgt-subsys-name> --skip-health-check
```

Some containers in the `kube-system` namespace show a status of `ErrImageNeverPull`

When upgrading, some of the containers in the **kube-system** namespace might show a status of**ErrImageNeverPull**. This happens due to Docker not successfully loading all images from the upgrade .tgz file. To resolve this issue and enable the upgrade to proceed, complete the following steps:

1. Run the following command to determine which node is missing the control plane files:

   ```
   kubectl -n kube-system get pods -owide
   ```

   This command returns the names of the nodes containing pods with the **ErrImageNeverPull** status, which indicates missing control plane files.

2. On the node that is missing the control plane files, run the following command to determine which versions of the control plane are missing:

   ```
   cat /var/lib/apiconnect/appliance-control-plane-current
   ```

3. V10.0.7: For each missing control plane, run the following command on the same node to add it, replacing *<version>* with the version of the control plane:

   ```
   ctr -k8s.io images import /usr/local/lib/appliance-control-plane/<version>/kubernetes.tar --all-platforms
   ```

   If you have a kubernetes.tgz file instead of a kubernetes.tar file, the run this first:

   ```
   gunzip /usr/local/lib/appliance-control-plane/<version>/kubernetes.tgz
   ```

4. V10.0.6: For each missing control plane, run the following command on the same node to add it, replacing *<version>* with the version of the control plane:

   ```
   docker load < /usr/local/lib/appliance-control-plane/<version>/kubernetes.tgz
   ```

Skipping health check when re-running upgrade

The **apicup subsys install** command automatically runs **apicup health-check** prior to attempting the upgrade. An error is displayed if a problem is found that will prevent successful upgrade

In some scenarios, if you encounter an upgrade failure, an attempt to rerun **apicup subsys install** is blocked by errors found by **apicup health-check**. Even when you have fixed the error (such as reconfiguration of an incorrect upgrade CR), the failed upgrade can continue to cause the health check to fail.

You can workaround the problem by adding the **--skip-health-check** flag to suppress the health check:

```
apicup subsys install <subsystem_name> --skip-health-check
```

In this case, use of **--skip-health-check** allows the upgrade to rerun successfully.

Unexpected behavior in Cloud Manager and API Manager UIs after upgrade

Stale browser cache issues can cause problems after an upgrade. To remedy this problem, clear your browser cache, and open a new browser window.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

- **Troubleshooting upgrades to v10.0.6.0 on VMware**
  Review the following troubleshooting guidance if you encounter a problem while upgrading to API Connect 10.0.6.0 on VMware.

Troubleshooting upgrades to v10.0.6.0 on VMware

Review the following troubleshooting guidance if you encounter a problem while upgrading to API Connect 10.0.6.0 on VMware.

In addition to the issues noted in Troubleshooting upgrades on VMware, you might encounter the following issues when upgrading specifically to version 10.0.6.0. These issues do not apply when upgrading to v10.0.7.0 or later.

The original primary PVC is not attached to any pods in this namespace

Upgrading to v10.0.6.0 only:

If the **apicops version:pre-upgrade** check reports

```
The original primary PVC is not attached to any pods in this namespace
```

and you have not started the upgrade yet (you have not run the **apicup subsys install** command) then follow these steps:

1. Take a backup of your management subsystem: Backing up and restoring the Management subsystem V10.0.6.0.
2. Restore the new backup: Restoring the management subsystem V10.0.6.0.

The action of taking and restoring a management backup fixes the problem that causes the error message. If you see this error after starting upgrade, see Original PostgreSQL primary not found.

Original PostgreSQL primary not found

Upgrading to v10.0.6.0 only:

If you ran the **apicup subsys install <tar file>** command to start an upgrade, the upgrade does not complete, and the **apicup health-check** command returns:

```
apicup subsys health-check management
Error: Cluster not in good health:
expect member apicdev.ibm.com 'Install stage' to be 'Done'(actual: SETUP_SUBSYSTEM) | Detail: reconcile.go#apply subsystem not
ready
```

then SSH into the management appliance and switch to root user to get more details of the error:

```
ssh apicadm@<management appliance>
sudo -i
```

Run **kubectl describe mgmt**, and check the output for the following message:

```
kubectl describe mgmt

...
Original PostgreSQL primary not found, further operand upgrade will be blocked. Set apiconnect-operator/db-primary-not-found-
allow-upgrade=true annotation to unblock operand upgrade and perform db backup and restore after upgrade to restore original
primary
...
```

To fix this problem, complete the following steps:

1. Create an extra-values.yaml file and set the annotation:

   ```
   metadata:
     annotations:
       apiconnect-operator/db-primary-not-found-allow-upgrade: "true"
   ```

   Note: If you already have an extra-values.yaml file, update the existing file with above annotation.
2. Set the extra-values-file property on your management subsystem with **apicup**:

```
apicup subsys set <management subsystem name> extra-values-file=<extra values filename>
```

3. Restart the upgrade, with the **--skip-health-check** flag:

```
apicup subsys install <management subsystem name> <upgrade tar file> <control plane file - if required> --skip-health-check
```

4. When upgrade is complete:
   a. Remove the annotation from your extra-values.yaml file.
   b. Take a backup of your management subsystem: Backing up and restoring the Management subsystem V10.0.6.0.
   c. Restore the new backup: Restoring the management subsystem V10.0.6.0. The action of taking and restoring a management backup fixes the problem that causes the error message `The original primary PVC is not attached to any pods in this namespace`. Be careful to restore from the backup that is taken after the upgrade, and not from a backup taken before upgrade.

The `cloud-final.service` fails during upgrade

Upgrading to v10.0.6.0 only:

Sometimes during upgrade, there is an issue with the **cloud-final.service** and a node, and the appliance-manager enters a bad state.

To determine whether your upgrade encountered this problem, complete the following steps:

1. Check the output of **apic health-check** command for a result similar to the following example:

```
# apic health-check
INFO[0000] Log level: info
FATA[0000] Unable to cluster status: rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing dial tcp 9.20.153.38:9178: connect: connection refuse
```

2. Check the response to the **journalctl -u appliance-manager | grep cloud-final** command and see if it looks like the following example:

```
# journalctl -u appliance-manager | grep cloud-final
Nov 21 19:41:58 apimdev1040 apic[2569]: Job for cloud-final.service failed because the control process exited with error code.
Nov 21 19:41:58 apimdev1040 apic[2569]: See "systemctl status cloud-final.service" and "journalctl -xe" for details.
```

If you determined that this was the problem, you can correct it by running each of the following commands on every node:

1. **systemctl restart cloud-final.service**
2. **apic lock**
3. **apic unlock**

Postgres replica fails to start due to permissions issue

Upgrading to v10.0.6.0 only:

Symptom:

```
2023-04-11 11:31:40,535 INFO: Lock owner: v10-0-5-up-ce0f24dc-site1-postgres-iszz-c5d9c4f5c-9d9fj; I am v10-0-5-up-ce0f24dc-site1-postgres-75845474bd-kzcjl
2023-04-11 11:31:40,537 INFO: starting as a secondary
2023-04-11 11:31:41.032 UTC [151610][]FATAL:  data directory "/pgdata/v10-0-5-up-ce0f24dc-site1-postgres" has invalid permissions
2023-04-11 11:31:41.032 UTC [151610][]DETAIL:  Permissions should be u=rwx (0700) or u=rwx,g=rx (0750).
2023-04-11 11:31:41,055 INFO: postmaster pid=151610
/tmp:5432 - no response
2023-04-11 11:31:41,091 INFO: Lock owner: v10-0-5-up-ce0f24dc-site1-postgres-iszz-c5d9c4f5c-9d9fj; I am v10-0-5-up-ce0f24dc-site1-postgres-75845474bd-kzcjl
2023-04-11 11:31:41,092 INFO: failed to start postgres
```

Resolve the issue by completing the following steps to correct the permissions and restart the replica pod:

1. Run the following command to exec into the pod:

```
k exec -it <pg-primary-pod> -- bash
```

2. Run the following command to reset the permissions:

```
chmod 0700 /pgdata/<pg-cluster-name>
```

3. Restart the failed pod.

Upgrade stuck with "Unable to upgrade appliance-base: exit status 100" message

Upgrading to v10.0.6.0 only:

This error can occur when the deployment is using an incorrect version of **containerd**. There is a known issue where upgrades from 10.0.1.7-eus or 10.0.1.8-eus to 10.0.5.1 incorrectly upgraded the **containerd** version and subsequent upgrades from 10.0.5.1 might encounter this error.

On each node, complete the following steps to determine the version of **containerd** and if necessary, downgrade it to the correct version:

1. Run the following set of commands on the healthy node to downgrade the version of **containerd**:

```
    systemctl stop appliance-manager
    systemctl stop kubelet
    systemctl stop containerd
    apt-get --allow-downgrades upgrade -y containerd.io
```

```
sed -i 's/KillMode=process/KillMode=mixed/g' /lib/systemd/system/containerd.service
systemctl daemon-reload
systemctl restart containerd
apic lock
apic unlock
```

2. Run the following command to verify that `containerd` now displays `containerd://1.5.11` for the CONTAINER-RUNTIME:

```
kubectl get nodes -o wide
```

Remember to complete these steps on every node.

Pod stuck in `Pending` status during upgrade

Upgrading to v10.0.6.0 only:

When upgrading, the scheduler might deploy a subset of the same microservice pods on the same node. This can prevent other pods with **requiredDuringSchedulingIgnoredDuringExecution** affinity rules from being deployed due to a lack of resources on a subset of nodes. To allow the pending containers to be deployed successfully, identify any pods of the same type that are scheduled on the same node and delete one of them. This will free up space and cause the deleted pod to get rescheduled. To identify pods that are eligible to be deleted, and then delete the pods, complete the following steps:

1. Run the following command and check for any pods of the same type that are on the same node:

```
kubectl get po -o=custom-columns='name:.metadata.name, node:.spec.nodeName,
antiaffinity:.spec.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution' | grep -v '<none>' | awk
'{print $1" "$2'}
```

   In the following example snippet, one of the two **apim** pods on the node test0186 should be deleted:

```
stv3-management-analytics-proxy-56848c8c69-phpdh test0186
stv3-management-analytics-proxy-56848c8c69-sc45f test0187
stv3-management-analytics-proxy-56848c8c69-scf6g test0188
stv3-management-apim-5574796948-6lnwj test0186
stv3-management-apim-5574796948-h9dgx test0186
stv3-management-apim-5574796948-tsb4g test0188
```

2. Run the following command to delete a pod:

```
kubectl delete po <pod_name>
```

Database replica pods stuck in Unknown or Pending state

Upgrading to v10.0.6.0 only:

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **Unknown** or a **Pending** state after a number of minutes. The pod fail to get into a **Running** state.

This situation occurs when the replicas do not initialize properly. You can use the **patronictl reinit** command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. SSH into the VM as root.
2. Exec onto the failing pod:

```
kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

3. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
|                          Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
+----------------------------------------------------------+----------------+--------+--------------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | start failed |    |   unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s  | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m  |  172.16.53.68  |        |   running    |  3 |         0 |
+----------------------------------------------------------+----------------+--------+--------------+----+-----------+
```

   In the example shown above **fxpk-management-01191b80-postgres-586f899fdf-6s25b** is not in running state.

   Note the **clusterName** and **replicaName** which are not up:

   - **clusterName** - **fxpk-management-01191b80-postgres**
   - **replicaName** - **fxpk-management-01191b80-postgres-586f899fdf-6s25b**

4. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

   Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
|                          Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
+----------------------------------------------------------+----------------+--------+--------------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | start failed |    |   unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s  | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m  |  172.16.53.68  |        |   running    |  3 |         0 |
+----------------------------------------------------------+----------------+--------+--------------+----+-----------+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

5. Run `patronictl list` again.
   You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

   For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+---------+---------+----+-----------+
|                         Member                         |      Host      |  Role   |  State  | TL | Lag in MB |
+--------------------------------------------------------+----------------+---------+---------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b  | 172.16.172.244 |         | running |  1 |     23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader | running |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |         | running |  3 |         0 |
+--------------------------------------------------------+----------------+---------+---------+----+-----------+
```

6. The pod that previously was in an `Unknown` or `Pending state` or `(0/1) Running` state is now in `(1/1) Running` state.

The `etcd` pod is stuck in `Terminating` state

Upgrading to v10.0.6.0 only:

During an upgrade, you might see a health-check reporting the upgrade is stuck in `ETCD` stage and that an `etcd` pod is stuck in the `Terminating` state.

Verify the issue by completing the following steps:

1. SSH into the VM as root.
2. Run the following command and verify that the upgrade stage is `ETCD`:

   ```
   apic status
   ```

3. Run the following command to determine whether an `etcd` pod is stuck in the `Terminating` state:

   ```
   kubectl get pods -n namespace | grep etcd
   ```

Resolve the issue by completing the following steps:

1. Run the following command to retrieve the names of the `etcd` pods:

   ```
   kubectl get pods -n namespace -o wide | grep etcd
   ```

2. On the pod that is stuck, SSH into the VM as root.
3. Run the following command to restart the pod:

   ```
   systemctl restart kubelet
   ```

Maintaining API Connect

You can use utilities to complete maintenance tasks such as backup, restore, and certificate management in a VMware environment.

Note: When maintaining API Connect, do not use `kubectl exec` commands to access API Connect pods unless advised by IBM.
Do not make any changes on the deployed VMs unless documented here or otherwise advised by IBM. Attempting to manually update packages, adding new users, or installing new software will likely cause problems. Operating system updates are handled by API Connect fix packs.

- **Advanced installation on VMware**
  Optionally install additional software.
- **Advanced configuration on VMware**
  Perform configuration tasks on your API Connect deployment on VMware.
- **Advanced configuration for management subsystem**
  Specify advanced configuration for the management subsystem.
- **Configuring rate limits on the gateway nodes on VMware**
  Create an env.yaml file on each Gateway node to configure the default rate limit.
- **Backing up and restoring on VMware**
  Backup and restore API Connect on VMware.
- **Disaster recovery**
  Prepare for disaster events and recover API Connect on VMware if a disaster event occurs.
- **Disabling and re-enabling the analytics subsystem on VMware**
  Disable the analytics subsystem by shutting down the VM that hosts it.
- **Maintaining a two data center deployment**
  How to maintain a two data center disaster recovery (DR) deployment on VMware, including information about normal operation, failure handling, and recovery.
- **Monitoring and health checks**
  Monitor your deployment and complete health checks on subsystems.
- **Logging**
  Configure logging for your deployment on VMware.
- **Troubleshooting**
  Troubleshoot your deployment on VMware.

Advanced installation on VMware

Optionally install additional software.

See:

- **Installing the IBM License Metric Tool on VMware**
  The IBM® License Metric Tool (ILMT) helps you determine whether your IBM API Connect deployment is compliant with licensing requirements.
- **Installing the toolkit**
- **Installing the Automated API behavior testing application**
  Install the Automated API behavior testing application in your IBM API Connect deployment on VMware.

Installing the IBM License Metric Tool on VMware

The IBM® License Metric Tool (ILMT) helps you determine whether your IBM API Connect deployment is compliant with licensing requirements.

About this task

ILMT provides useful features for managing virtualized environments and measuring license utilization. ILMT discovers the software that is installed in your infrastructure, helps you to analyze the consumption data, and allows you to generate audit reports. Each report provides you with different information about your infrastructure; for example the computer groups, software installations, and the content of your software catalog.

By default, every ILMT audit report presents data from the previous 90 days. You can customize the type and amount of information displayed in a report by using filters, and save your personal settings for future use. You can also export the reports to .csv or .pdf format, and schedule report emails so that specified recipients are notified when important events occur.

To deploy the ILMT with API Connect, complete the following steps for the Management, Developer Portal, and Analytics OVAs. The instructions refer to topics in the ILMT documentation:

Procedure

1. Add a disconnected data source to your ILMT deployment as explained in Adding a data source.
2. Download the scanner installer as explained in Downloading the disconnected scanner package (disconnected scenario).
3. Install the scanner and run scans as explained in Installing the disconnected scanner and gathering scan results (disconnected scenario).
4. Set up daily transfer of scan results to ILMT as explained in Setting up daily transfer of scan results to the License Metric Tool server (disconnected scenario).

Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just to need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

Procedure

To install and run the toolkit, complete the following steps:

1. Download the toolkit compressed file.
   - To download the toolkit from IBM Fix Central, complete the following steps:
     - Open the IBM Fix Central site in your browser.
     - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
     - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
     - In the Text field, enter `toolkit`, then click Continue.
     - Select the toolkit file that you want to download.
     - Click Continue, then follow the instructions to complete the download operation.
   - To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
     - Open the Cloud Manager or API Manager user interface.
     - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
     - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
     - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the designer_credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 6.

- To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 5.
- Close the Install API Connect Toolkit window.

2. Extract the contents of the toolkit compressed file to a folder of your choice.
   The contents of the file depend on the your chosen toolkit option and platform, as follows:

   - The apic-slim or apic-slim.exe file is the CLI for IBM API Connect.
   - The apic or apic.exe file is the CLI for IBM API Connect including LoopBack support.
     Tip: If you are using the CLI option, then if you rename the apic-slim file to apic, or the apic-slim.exe file to apic.exe, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
   - The api_designer-*platform* file is the API Designer user interface application for the specified platform.

3. Optional: Delete the **`$HOME/.apiconnect/node_clis`** directory.
   You need to do this only if you replaced a version of **`apic`** or apic.exe and are using the **`apic lb4`** command. You need to delete the directory get **`apic`** to unpack the new loopback.

   On Windows, $HOME is defined by environment USERPROFILE.

4. Run the CLI.
   - For the Mac OSX or Linux® platforms, complete the following steps:
     - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Make the CLI file an executable file by entering the following command:

       `chmod +x download_name`

       Where *download_name* is the name of the toolkit file that you downloaded, either apic or apic-slim.
     - Run CLI commands as follows:

       `./apic command_name_and_parameters`

       or

       `./apic-slim command_name_and_parameters`

       For details of the CLI commands, see apic.
   - For the Windows platform, complete the following steps:
     - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Run CLI commands as follows:

       `apic command_name_and_parameters`

       or

       `apic-slim command_name_and_parameters`

       For details of the CLI commands, see apic.

   Tip: Add the folder location of your CLI file to your PATH variable so that you can run CLI commands from anywhere in your file system.

5. Install the toolkit credentials.
   If you do not install the toolkit credentials that are provided for download, as detailed previously in the toolkit credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

   `apic client-creds:set toolkit_credentials_file_path/credentials.json`

   where *credentials_file_path* is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.
   Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.
   To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

   `apic client-creds:clear`

   For increased security, an administrator can remove the default credentials from the management server by completing the following steps:
   a. Log in to the management server as an administrator; see Logging in to a management server.
   b. Delete the default credentials by running the following commands:

   ```
   apic registrations:delete toolkit --server mgmt_endpoint_url
   apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
   ```

6. Install the API Designer credentials.
   If you do not install the API Designer credentials that are provided for download, as detailed previously in the API Designer credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the **`APIC_DESIGNER_CREDENTIALS`** environment variable to the credentials download location, using the mechanism appropriate for your operating system.

   After you have set the **`APIC_DESIGNER_CREDENTIALS`** environment variable, your API Designer uses the new credentials to authenticate with the management server.

   - Windows:
     If you are using Windows, create an environment variable using one of the following methods:

     - Create a permanent environment variable so that you can start API Designer from any location on your computer:
       - Open the Environment Variables page: Click Start > Settings > System and in the "Related Settings" section of the page, click Advanced System Settings.
       - On the Advanced tab of the System Properties dialog box, click Environment Variables.
       - In the "User variables" section, click New and create an environment variables with the following settings:
         - Variable: **`APIC_DESIGNER_CREDENTIALS`**

- Value: *<designer_credentials_file_path>*\designer_credentials.json where **`<designer_credentials_file_path>`** is the location where you stored the designer_credentials.json file.
- Click OK to save the new environment variable, and then exit the dialog box.

- If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
  - Open the Windows command prompt.
  - Run the following command to set the temporary environment variable:

    ```
    set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
    ```

    where **`<designer_credentials_file_path>`** is the location where you stored the designer_credentials.json file.

    Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.
  - Now run the following command to start API Designer:

    ```
    C:\"Program Files\API Designer\API Designer.exe"
    ```

    By default, API Designer is installed in the C:\"Program Files\API Designer folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

- Mac:
  If you are using Mac OS X, create an environment variable using one of the following methods:

  - Run the following command to set the global environment variable:

    ```
    launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
    ```

    where **`<designer_credentials_file_path>`** is the location where you stored the API Designer credentials JSON file.
  - Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

    ```
    APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open
    <designer_application_file_path>/'API Designer.app'
    ```

    where:
    - **`<designer_credentials_file_path>`** is the location to which you downloaded the API Designer credentials JSON file.
    - **`<designer_application_file_path>`** is the location to which you downloaded and uncompressed the API Designer application.

7. Start the API Designer application from the location to where you stored the extracted file.
   Note:
   - To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
     - In Windows File Explorer, navigate to the USER_HOME\AppData\Local\Programs\api-designer folder.
     - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.
   - To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see Developing your APIs and applications.

For information on using the toolkit CLI, see Using the developer toolkit command-line tool.

Installing the Automated API behavior testing application

Install the Automated API behavior testing application in your IBM® API Connect deployment on VMware.

About this task

Use the following steps to install the Automated API behavior testing application as part of your initial deployment, or to add it in later.
Note: It is not possible to use the Automated API behavior testing application with a two data center disaster recovery configuration (Two data center warm-standby deployment strategy on VMware).

Procedure

1. Run the following **`apicup`** commands to install or update your deployment with the the Automated API behavior testing application:

   ```
   apicup subsys set mgmt test-and-monitor-enabled true
   apicup certs set mgmt hub --clear
   apicup certs set mgmt turnstile --clear
   apicup subsys set mgmt hub hub-endpoint
   apicup subsys set mgmt turnstile turnstile-endpoint
   apicup subsys install mgmt
   ```

   where:
   - **`mgmt`** is the name of your Management server (which you are either installing or updating).
   - **`hub-endpoint`** is an endpoint on the ingress subdomain for the API Connect stack; for example: hub.testsrv0231.subnet1.example.com.
   - **`turnstile-endpoint`** is an endpoint on the ingress subdomain for the API Connect stack; for example: turnstile.testsrv0231.subnet1.example.com.

   Attention: The hub and turnstile endpoints should be distinct and must not overlap with any of the 4 management endpoints.

2. Verify that the application deployed properly by completing the following steps:
    a. Log in to API Manager.
    b. On the Home page, click the Test APIs tile.
    c. Verify that the Automated API behavior testing opens.

Advanced configuration on VMware

Perform configuration tasks on your API Connect deployment on VMware.

See:

- **Adding a static route on a virtual machine**
  You can add a static route to the routing table when deploying API Connect on a VMware virtual machine.
- **Adding disk space to a VMware appliance**
  Increase disk space on the VMware appliance.
- **Appliance boot mode configuration**
  Configure the boot mode for the appliance VMs.
- **Running commands at boot with iso-ignored boot mode**
  When `iso-ignored` boot mode is configured, complete the following steps on each appliance that should run commands at boot.
- **Changing deployment profiles on VMware**
  You can change an API Connect installation to use a different deployment profile.
- **Changing the DNS server configuration on appliances**
  Complete this task only if you must change the DNS addresses after the initial boot of the VM.
- **Configuring API Connect subsystems in a cluster**
  This topic describes how to configure a cluster of API Connect subsystems (management server, analytics, and Developer Portal) with three VMs for each subsystem, for use with a load balancer, to support a high availability (HA) environment.
- **Configuring SSHD to limit access to your deployment**
  How to configure SSHD in order to limit host access to your VMware deployment, or to change the SSHD cipher level.
- **Enabling Developer Portal feature flags on VMware**
  Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.
- **Managing an appliance data disk**
  You can use `apic` commands to manage appliance data disks in your VMware deployment.
- **Replica override for microservices on VMware**
  Modify the replica count for microservices by using template overrides for Custom Resources.
- **Testing a new configuration against a current cluster**
  You can use the `apicup` utility to test the validity of a new configuration before you install it.
- **TLS certificate concepts and operations**
  Information about internal API Connect certificates, renewing expired certificates, exporting certificates, and other certificate maintenance operations.
- **Use JWT security instead of mTLS between subsystems**
  If your network requires TLS termination on load-balancers located between your API Connect subsystems, then disable mTLS and enable JWT to provide application layer security.
- **Credentials and encryption secrets reference**
  API Connect user credentials and database encryption secrets that are managed with the **apicup certs** command.

Adding a static route on a virtual machine

You can add a static route to the routing table when deploying API Connect on a VMware virtual machine.

About this task

Add commands to your additional `cloud-init` file. You can do this for the Management, Analytics, and Portal subsystems.

Procedure

1. Specify an additional cloud-init file:

   ```
   apicup subsys set <subsys> additional-cloud-init-file <path-to-cloud-init-file>
   ```

2. Add lines to your cloud-init file.
   Syntax:

   ```
   bootcmd:
    - ip route add <destination>/<mask> via <gateway> dev eth<n>
   ```

   Example:

   ```
   bootcmd:
    - ip route add 172.27.218.0/24 via 172.27.218.1 dev eth1
    - ip route add 172.26.203.0/24 via 172.27.218.1 dev eth1
   ```

3. Regenerate the ISO file.

   ```
   apicup subsys install <subsys> --out <plan-directory>
   ```

4. Restart the ISO, and ensure that the node(s) have this updated ISO attached at startup.

Adding disk space to a VMware appliance

Increase disk space on the VMware appliance.

About this task

For multi-node deployment of the appliance, only one VM at a time should have its disk space augmented. The subsystem should be in good health before proceeding to the next node. Health of the subsystem can be checked with **`apicup subsys health-check <subsystem-name>`**.

Restriction:
The increased disk capacity of a VMware appliance won't be reflected in the storage size of the PVCs that are hosted on the appliance, as those PVCs are immutable. This restriction doesn't impact the Portal and Analytics subsystems, as the PVC storage size isn't used to enforce any disk usage limit, so the subsystems can use all of the additional disk space. However, for the Management subsystem the PVC storage size is used in the database storage utilization computation, and that might trigger disk usage warnings.

The most common reason for the Management subsystem database to run out of disk space, is due to Write Ahead Log (WAL) file accumulation in relation to an incorrectly configured backup, or an unavailable backup server. If this type of condition arises, adding disk space might help with the health of the appliance control plane, but it won't increase the disk space that's available to the database, so you must resolve the WAL file accumulation condition. If the disk space that's available to the Management database must be increased, you should carry out a disaster recovery operation of the Management subsystem, using a larger data disk size for the new installation, so that adding disk space after the install won't be needed.

Procedure

1. Shutdown the virtual machine, either by issuing a shutdown command as root from the VM, or by using the VMware console or command line interface.
   For example, use the VMware console user interface and click the red rectangle.

2. Use either the VMware console or command line interface to add a new device to the VM, of type Hard Disk and of the desired size to be added to the appliance.
   For example, on the VMware console:
   a. Click Edit settings.
   b. Click Add new device. From the drop-down menu, select Hard Disk.
   c. Enter size in GB and click OK.
3. Use either the VMware console or the command line interface to power on the VM.
   As the VM starts it should detect the newly added disk and use it to augment the available capacity for the /data/secure mount point.
4. Optionally you may verify that a new disk **`sdc`** was added (or **`sdd`** if you already added one disk, and so on). Log in to the server and observe the new disk. For example:

```
root@apimdev1139:~# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                   8:0    0  100G  0 disk
├─sda1                8:1    0 99.9G  0 part  /
├─sda14               8:14   0    4M  0 part
└─sda15               8:15   0  106M  0 part  /boot/efi
sdb                   8:16   0  250G  0 disk
└─apiconnect-data   252:0    0  260G  0 lvm
  └─apicSecureDisk  252:1    0  260G  0 crypt /data/secure
sdc                   8:32   0   10G  0 disk
└─apiconnect-data   252:0    0  260G  0 lvm
  └─apicSecureDisk  252:1    0  260G  0 crypt /data/secure
sr0                  11:0    1   44K  0 rom
```

Appliance boot mode configuration

Configure the boot mode for the appliance VMs.

By default every appliance node requires the presence of an ISO file attached at boot.

Version 10.0.4.0 or later: You can configure individual VMs so that they can be booted and rebooted without an ISO being attached. Each appliance VM (node) is configured individually.

You can:

- Check the current boot mode of an appliance
- Configure an appliance for iso-ignored boot mode
- Configure an appliance for iso-required boot mode
- Revert back to iso-required boot mode by booting the appliance in rescue mode

Check the current boot mode of an appliance

The default boot mode for all appliances is **`iso-required`**. To check the current boot mode:

1. Connect to the target appliance via SSH then switch to the root user:

```
ssh apicadm@{ova appliance hostname}
sudo -i
```

2. Use **`apic boot-mode`** to check the current boot mode of the appliance. For example:

```
root@ip-192-168-122-241:~# apic boot-mode
INFO[0000] Log level: info
```

```
INFO[0000] current boot mode is: iso-required
```

Configure an appliance for `iso-ignored` boot mode

An appliance can be configured to use the **iso-ignored** boot mode instead of the default **iso-required**. Prerequisite: The appliance must have first booted with its configuration ISO attached and in **iso-required** boot mode (default) before the boot mode can be changed to **iso-ignored**.

You can configure an appliance to use he **iso-ignored** boot mode instead of the default **iso-required**.

Prerequisite: The appliance must have first booted with its configuration ISO attached and in **iso-required** boot mode (default), before you can change the boot mode to **iso-ignored**.

Note: If you added a static route to the routing table, as documented in Adding a static route on a virtual machine, and if you are relying on a **bootcmd** from the ISO, then using the **iso-ignored** boot mode causes the **bootcmd** section being ignored or ineffective.

1. Connect to the target appliance via SSH then switch to the root user:

   ```
   ssh apicadm@{ova appliance hostname}
   sudo -i
   ```

2. Edit the file **/etc/default/grub.d/50-cloudimg-settings.cfg** to ensure that the **GRUB_TIMEOUT** and **GRUB_RECORDFAIL_TIMEOUT** are set to the value **5**. Then run the command **update-grub**. For example:

   ```
   root@ip-192-168-122-241:~# update-grub
   Sourcing file `/etc/default/grub'
   Sourcing file `/etc/default/grub.d/50-cloudimg-settings.cfg'
   Sourcing file `/etc/default/grub.d/init-select.cfg'
   Generating grub configuration file ...
   Found linux image: /boot/vmlinuz-5.4.0-84-generic
   Found initrd image: /boot/initrd.img-5.4.0-84-generic
   done
   ```

3. Set the appliance boot mode to **iso-ignored** with the command **apic boot-mode iso-ignored**. For example:

   ```
   root@ip-192-168-122-241:~# apic boot-mode iso-ignored
   INFO[0000] Log level: info
   INFO[0000] current boot mode is: iso-ignored
   ```

4. Shutdown the VM, and remove the CD/DVD device from the VM configuration through VSphere.
   For any subsequent boot of the appliance while it remains in **iso-ignored** boot mode, any ISO file attached will be ignored, and the boot will proceed as normal, whether an ISO file is attached or not.

   Warning: If the appliance was booted in **iso-ignored** boot mode, it is possible to change its boot mode to **iso-required**. However, as stated in the prerequisites, it is not possible to immediately switch back to **iso-ignored**, and the next boot of the appliance must happen with the configuration ISO attached and boot mode **iso-required**.

Configure an appliance for `iso-required` boot mode

The mode **iso-required** is the default boot mode for all appliances. If the appliance boot mode was changed to **iso-ignored** you can revert back to **iso-required** boot mode.

Warning:
If the appliance was booted in **iso-ignored** boot mode, it is possible to change its boot mode to **iso-required**. However, you cannot immediately switch back to **iso-ignored**, and the next boot of the appliance must happen with the configuration ISO attached and boot mode **iso-required**. After the appliance is booted with ISO attached in **iso-required** boot mode, the boot mode can then be switched to **iso-ignored** if needed.

1. Connect to the target appliance via SSH then switch to the root user:

   ```
   ssh apicadm@{ova appliance hostname}
   sudo -i
   ```

2. Set the appliance boot mode to **iso-required** with the command **apic boot-mode iso-required**. For example:

   ```
   root@ip-192-168-122-241:~# apic boot-mode iso-required
   INFO[0000] Log level: info
   INFO[0000] current boot mode is: iso-required
   ```

   Important: For any subsequent boot of the appliance while it remains in **iso-required** boot mode, the configuration ISO file must be connected at boot and for every boot.

Revert back to `iso-required` boot mode by booting the appliance in rescue mode

When an appliance is configured with the **iso-ignored** boot mode, the boot sequence ignores any attached ISO. This means that it is not possible to influence the appliance configuration during its boot. In case the appliance is not booting properly and the boot mode needs to be reverted back to **iso-required**, proceed as follows:

1. Access to the VMware console for the Appliance, and after causing it to reboot, bring up a console to the appliance. You can give focus to the window by clicking the mouse into it. Press the **Esc** key to access the Linux grub boot menu. For example:

   ```
   GNU GRUB version 2.04

   *Ubuntu
   Advanced options for Ubuntu
   ```

2. Press **e** and navigate using arrow keys to the line starting with **linux** to append **systemd.unit=rescue.target**

```
GNU GRUB version 2.04
.
.
.
insmod part_gpt
insmod ext2
if [ x$feature_platform_search_hint = xy ];  then
   search --no-floppy --fs-uuid --set=root e46da1b8-55ao-4df9-842e-\
     8d80c3a22ffc
else
   search --no-floppy --fs-uuid --set=root e46da1b8-55ao-4df9-842e-8\
     d80c3a22ffc
fi
linux      /boot/vmlinuz-5.4.0-84-generic root=LABEL=cloudimg-roo\
   tfs ro console=tty1 console=ttyS0 net.ifnames=0 bioosdevname=0 systemd.unit\
   =rescue.target
initrd     /boot/initrd.img-5.4.0-84-generic
```

3. Press **Ctrl+x** to boot.
4. Press **Enter** to enter maintenance mode. Run the commands:

   ```
   rm -f /etc/cloud/cloud-init.disabled
   ```

   and

   ```
   echo -n "iso-required" > /usr/local/lib/appliance-config/boot-mode
   ```

   ```
   [   3.442687] systemd[1]: Mounted Kernel Debug Filesystem
   [   3.444253] systemd[1]: Mounted Kernel Trace Filesystem
   [   3.445680] systemd[1]: Finished Create list of static device nodes for the current kernel
   [   3.446777] systemd[1]: Finished Set console scheme.
   [   3.449347] EXT4-fs (sda1): re-mounted. Opts: (null)
   [   3.452687] systemd[1]: Finished Remount Root and Kernel File Systems
   [   3.455123] systemd[1]: Finished Load Kernel Modules.
   [   3.458487] systemd[1]: Mounting FUSE Control File System...
   [   3.461333] systemd[1]: Mounting Kernel Configuration File System...
   [   3.462687] systemd[1]: Condition check resulted in Rebuild Hardware Database being skipped.
   [   3.464498] systemd[1]: Condition check resulted in Platform Persistant Storage Archival being skipped.
   [   3.466292] systemd[1]: Starting Load/Save Randome Seed...
   [   3.467324] systemd[1]: Starting Apply Kernel Variables...
   [   3.470531] systemd[1]: Starting Create System Users...
   [   3.471847] systemd[1]: Mounted FUSE Control File System
   [   3.472964] systemd[1]: Started Journal Service
   Press Enter for maintenance:
   (or press Ctrl-D to continue):
   root#ip-192-168-122-241:~# rm -f /etc/cloud/cloud-init.disabled
   root#ip-192-168-122-241:~# echo -n "iso-required" > /usr/local/lib/appliance-config/boot-mode
   root#ip-192-168-122-241:~#
   ```

5. Reboot the appliance with correct configuration ISO attached.

Running commands at boot with `iso-ignored` boot mode

When **iso-ignored** boot mode is configured, complete the following steps on each appliance that should run commands at boot.

About this task

If you configured the appliance to boot in **iso-ignored** mode as documented in Appliance boot mode configuration, the **bootcmd** lines will be ignored. For each appliance node that should run commands at boot, complete the following steps.

Procedure

1. SSH into the appliance as root.
2. Create the /usr/local/bin/appliance-boot-commands.sh script to be executed at boot; for example:

   ```
   cat << EOF > /usr/local/bin/appliance-boot-commands.sh
   #!/bin/bash

   ip route add 172.27.218.0/24 via 192.168.122.1 dev eth0
   ip route add 172.26.203.0/24 via 192.168.122.1 dev eth0
   EOF
   ```

   This step takes the same example, and essentially achieves the same result, as the procedure in Adding a static route on a virtual machine.

3. Make the script executable by running the following command:

   ```
   chmod +x /usr/local/bin/appliance-boot-commands.sh
   ```

4. Create the following systemd service file /lib/systemd/system/appliance-boot-commands.service:

   ```
   cat << EOF > /lib/systemd/system/appliance-boot-commands.service
   [Unit]
   After=network-online.target

   [Service]
   Type=oneshot
   ExecStart=/usr/local/bin/appliance-boot-commands.sh
   ```

```
RemainAfterExit=false
StandardOutput=journal

[Install]
WantedBy=multi-user.target
EOF
```

5. Enable the systemd service by running the following command:

   ```
   systemctl enable appliance-boot-commands
   ```

   Upon subsequent boots, the **/usr/local/bin/appliance-boot-command.sh** script will be executed.

Changing deployment profiles on VMware

You can change an API Connect installation to use a different deployment profile.

For information on the available deployment profiles, see Planning your deployment topology and profiles. The steps for changing deployment profile after installation depend on whether you want to change to a profile that has a different number of VMs and whether additional CPU and memory resources are required for the profile you move to.
Note: The procedures to change deployment profile that are described in this topic do not permit the endpoints to be changed. The API Connect endpoints are listed here: Deployment overview for endpoints. If you want to change a one replica deployment to a three replica deployment, and you used your VM hostnames in your endpoints, then you must configure load-balancers and logical DNS records so that your original VM hostnames route to your new VMs. Alternatively, you can change endpoints by following the form factor migration procedure: Migrating from v10 to v10 on a different form factor.

Moving to a profile that has the same number of VMs

If you are moving to a profile that requires greater CPU and memory resources, follow these steps:

1. Complete the disaster recovery preparation steps for your subsystems: Preparing for a disaster.
2. Shutdown your API Connect VMs.
3. Update the CPU and memory resources for your API Connect VMs in your VMware console, as required for your new profile.
4. Restart your VMs.
5. In your project directory, update your deployment profile:

   ```
   apicup subsys set <subsystem_name> deployment-profile=<new profile>
   ```

6. Apply the new profile:

   ```
   apicup subsys install <subsystem_name>
   ```

If you are moving to a profile that requires lower CPU and memory resources, follow these steps:

1. Complete the disaster recovery preparation steps for your subsystems: Preparing for a disaster.
2. In your project directory, update your deployment profile:

   ```
   apicup subsys set <subsystem_name> deployment-profile=<new profile>
   ```

3. Apply the new profile:

   ```
   apicup subsys install <subsystem_name>
   ```

4. Shutdown your API Connect VMs.
5. Update the CPU and memory resources for your API Connect VMs in your VMware console, as required for your new profile.
6. Restart your VMs.

If for any reason you are not able to change the CPU or memory on your VMs, redeploy them following the disaster recovery process: Recovering from a disaster.

Moving from a one replica to a three replica deployment profile

1. Complete the disaster recovery preparation steps for your subsystems: Preparing for a disaster.
   Note: You do not need to take VMware snapshots, since your existing VMs are deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

   ```
   apicup subsys set <subsystem_name> deployment-profile=<three replica profile>
   ```

3. Define the hostname and network interfaces for the two new VMs in each of your subsystems. Use the `apicup hosts create` and `apicup iface create` commands. For more information, see the hosts and interface create steps in the installation instructions for each subsystem:
   - Configuring the Management subsystem.
   - Configuring the Analytics subsystem.
   - Configuring the Developer Portal subsystem.
4. Shutdown your existing API Connect VMs.
   Note: To minimize downtime, you can delay shutting down your existing deployment until you are about to deploy their replacement VMs in the next step.
5. Follow the disaster recovery steps for each subsystem: Recovering from a disaster, but note that you must create new ISO files. Do not attempt to reuse the previous ISO file for one of the VMs in your new three replica deployment.

Moving from a three replica to a one replica deployment profile

1. Complete the disaster recovery preparation steps for your subsystems: Preparing for a disaster.
   Note: You do not need to take VMware snapshots, since your existing VMs are deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem_name> deployment-profile=<one replica profile>
```

3. Delete two of your three VMs from your configuration with the `apicup hosts delete` command.
4. Shutdown your existing API Connect VMs.
   Note: To minimize downtime, you can delay shutting down your existing deployment until you are about to deploy their replacement VMs in the next step.
5. Follow the disaster recovery steps for each subsystem: Recovering from a disaster, but note that you must create new ISO files. Do not attempt to reuse any previous ISO files for your new one replica deployment.

Changing the DNS server configuration on appliances

Complete this task only if you must change the DNS addresses after the initial boot of the VM.

Before you begin

Verify that you have VMware console access to the VM, as well as the correct password. If you cannot **ssh** into the appliance, you can use the VMware console interactive login instead (use the **apicadm** login and the password configured as **default-password** in the **apicup** project).

About this task

This task is only valid for making changes to DNS resolution. The IP address of the VM cannot be changed at any time after the initial boot.

Procedure

1. Edit the /etc/netplan/50-cloud-init.yaml file and update the list of nameserver addresses.
   Attention: Typically, you do not edit this file manually. You should only make direct changes to the /etc/netplan/50-cloud-init.yaml file when you need to update the nameserver addresses.
   The following example shows what the list of nameserver addresses looks like in the file:

```
network:
    ethernets:
        eth0:
            addresses:
            - 192.168.122.242/24
            nameservers:
                addresses:
                - 1.1.1.1
                - 1.0.0.1
                - 8.8.8.8
                search:
                - nip.io
            routes:
            -   metric: 10
                to: 0.0.0.0/0
                via: 192.168.122.1
    version: 2
```

2. Run the following command to apply the changes:

```
sudo netplan apply
```

3. Run the following command to verify the changes:

```
cat /run/systemd/resolve/resolv.conf
```

4. Attach an updated ISO to the VM to ensure all artifacts are up-to-date in case a disaster recovery is needed.
   a. Update the **apicup** project configuration with the correct DNS values.
   b. Generate a new ISO based on the updated project.
   c. Attach the new ISO to the VM.

Configuring API Connect subsystems in a cluster

This topic describes how to configure a cluster of API Connect subsystems (management server, analytics, and Developer Portal) with three VMs for each subsystem, for use with a load balancer, to support a high availability (HA) environment.

Before you begin

To create a cluster of hosts for each API Connect subsystem, use **apicup** to create a subsystem with all the required parameters, and to add as many hosts as needed to the configuration file. The configuration file is a .yml file in the project directory.

For each host of a subsystem added in the .yml file, a separate ISO file is created for the cluster member VM. Note that the ISO files for each VM must stay attached during the whole lifetime of the VM.

In an API Connect cluster in a VMware environment, the first three nodes are master nodes. Additional nodes are worker nodes.

Note: To add a new host to an existing cluster, you can create the host, regenerate the ISO and attach it to the new virtual machine, and it will automatically join the cluster.
To use these instructions, you should first review:

- Requirements for initial deployment on VMware
- Load balancer configuration in a VMware deployment
- Deployment overview for endpoints
- Firewall requirements on VMware
- Firewall enabled ports for clustered OVA deployments
- Important: For best performance it is recommended that the network latency between any 2 nodes be as low as possible. Do not configure nodes from the same subsystem cluster across multiple data centers with a high latency network. A high latency network is one that experiences more than 30ms latency between nodes.

About this task

This page presents a concise step-by-step flow, with sample commands, for the configuration of each subsystem. As you step through the flow, you might want to refer back to the detailed configuration steps for each subsystem. The detailed instructions provide additional considerations for each step, and include optional configuration tasks for backups, logging, message queues (analytics), and password hashing. Each of the subsystem pages describe how to use the VMware console to deploy the ISOs that you create here.

Detailed configuration steps:

- Deploying the Management virtual appliance
- Deploying the Analytics virtual appliance
- Deploying the Developer Portal virtual appliance

The example commands uses the following values for DNS server, internet gateway, host names and IP addresses:

Component	Host names and IP addresses
DNS Name Server	`IP: 192.168.1.1`
Internet gateway	`IP: 192.168.1.2`
Manager on VM1	`Hostname: manager1.sample.example.com` `IP: 192.168.1.101`
Manager on VM2	`Hostname: manager2.sample.example.com` `IP: 192.168.1.102`
Manager on VM3	`Hostname: manager3.sample.example.com` `IP: 192.168.1.103`
Analytics on VM4	`Hostname: analytics1.sample.example.com` `IP: 192.168.1.104`
Analytics on VM5	`Hostname: analytics2.sample.example.com` `IP: 192.168.1.105`
Analytics on VM6	`Hostname: analytics3.sample.example.com` `IP: 192.168.1.106`
Developer Portal on VM7	`Hostname: portal1.sample.example.com` `IP: 192.168.1.107`
Developer Portal on VM8	`Hostname: portal2.sample.example.com` `IP: 192.168.1.108`
Developer Portal on VM9	`Hostname: portal3.sample.example.com` `IP: 192.168.1.109`

Procedure

1. Create the Management subsystems
   a. Create the Management subsystem.

      ```
      apicup subsys create mgmt management
      ```

   b. Set deployment-profile to `n3xc4.m16`.

      ```
      apicup subsys set mgmt deployment-profile=n3xc4.m16
      ```

      If you omit this step, the deployment-profile defaults to `n1xc4.16`, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

   c. Specify the license version you purchased.
      `apicup subsys set` ***`mgmt`***
      `license-use=<`*`license_type`*`>`

      The *license_type* must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

   d. Set management endpoints
      Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following values point to a sample load balancer URL.

Component	Command
Management REST API URL	`apicup subsys set mgmt platform-api platform-api.sample.example.com`
Consumer (Portal) REST API URL	`apicup subsys set mgmt consumer-api consumer-api.sample.example.com`
Cloud Manager UI	`apicup subsys set mgmt cloud-admin-ui cloud-admin-ui.sample.example.com`
API Manager UI	`apicup subsys set mgmt api-manager-ui api-manager-ui.sample.example.com`

   e. Set the search domain for the VM.

      ```
      apicup subsys set mgmt search-domain sample.example.com
      ```

   f. Set the DNS Name Server for the VM to look up endpoints.

      ```
      apicup subsys set mgmt dns-servers 192.168.1.1
      ```

   g. Set a Public Keyfile.

This is the public key of the user account that you want to use to **ssh** from, to the appliance.

```
apicup subsys set mgmt ssh-keyfiles "id_rsa.pub"
```

h. Create the hosts for the subsystem.
You must specify a password to use to encrypt the disks that the appliance uses. Replace the example password in the previous example with a strong password that meets your security requirements.

```
apicup hosts create mgmt manager1.sample.example.com password123
apicup hosts create mgmt manager2.sample.example.com password123
apicup hosts create mgmt manager3.sample.example.com password123
```

i. Set the network interface. Note that the last parameter is the Internet Gateway.

```
apicup iface create mgmt manager1.sample.example.com eth0 192.168.1.101/255.255.255.0 192.168.1.2
apicup iface create mgmt manager2.sample.example.com eth0 192.168.1.102/255.255.255.0 192.168.1.2
apicup iface create mgmt manager3.sample.example.com eth0 192.168.1.103/255.255.255.0 192.168.1.2
```

j. Set the network traffic interfaces.

```
apicup subsys set mgmt traffic-iface eth0
apicup subsys set mgmt public-iface eth0
```

k. Verify the host configuration.

```
apicup hosts list mgmt
```

Note: This command might return the following messages, which you can ignore:

```
* host is missing traffic interface
* host is missing public interface
```

l. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use **apicup** to set the hashed password for your subsystem:

```
apicup subsys set mgmt default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.

m. Validate the installation.

```
apicup subsys get mgmt --validate
```

n. Create an ISO file in a plan folder. For example, **mgmtplan-out**.

```
apicup subsys install mgmt --out mgmtplan-out
```

If you have multiple nodes listed for the hosts, when you run the **--out** command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

o. To deploy the management ISOs on VMware, see Deploying the Management subsystem OVA file.

2. Create the analytics subsystems
   a. Create the subsystem:

```
apicup subsys create analyt analytics
```

   b. Set deployment-profile to **n3xc4.m16**.

```
apicup subsys set analyt deployment-profile=n3xc4.m16
```

   If you omit this step, the deployment-profile defaults to **n1xc2.16**, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

   c. Specify the license version you purchased.

```
apicup subsys set analyt
license-use=<license_type>
```

   The *license_type* must be either **production** or **nonproduction**. If not specified, the default value is **nonproduction**.

   d. Set analytics ingestion endpoint
   Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following value points to a sample load balancer URL.

Component	Command
analytics-ingestion	`apicup subsys set analyt analytics-ingestion=analytics-ingestion.sample.example.com`

   e. Set the search domain for the VM.

```
apicup subsys set analyt search-domain sample.example.com
```

   f. Set the DNS Name server for the VM to look up endpoints.

```
apicup subsys set analyt dns-servers 192.168.1.1
```

   g. Set a Public Keyfile.
   This is the public key of the user account that you want to use to **ssh** from, to the appliance

```
apicup subsys set analyt ssh-keyfiles "id_rsa.pub"
```

   h. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use **apicup** to set the hashed password for your subsystem:

```
apicup subsys set analyt default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.

i. Create the hosts for the subsystem. You must specify a password to use to encrypt the disks that the appliance uses. Replace the example password in the following password with a strong password that meets your security requirements.

```
apicup hosts create analyt analytics1.sample.example.com password123
apicup hosts create analyt analytics2.sample.example.com password123
apicup hosts create analyt analytics3.sample.example.com password123
```

j. Set the network interface.
Note that the last parameter is the Internet Gateway.

```
apicup iface create analyt analytics1.sample.example.com eth0 192.168.1.104/255.255.255.0 192.168.1.2
apicup iface create analyt analytics2.sample.example.com eth0 192.168.1.105/255.255.255.0 192.168.1.2
apicup iface create analyt analytics3.sample.example.com eth0 192.168.1.106/255.255.255.0 192.168.1.2
```

k. Check the host configuration for problems.

```
apicup hosts list analyt
```

l. Validate the installation.

```
apicup subsys get analyt --validate
```

m. Create an ISO file in a plan folder. For example, `analytplan-out`.

```
apicup subsys install analyt --out analytplan-out
```

If you have multiple nodes listed for the hosts, when you run the `--out` command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

n. To deploy the ISOs, see Deploying the Analytics subsystem OVA file.

3. Create the portal subsystems

a. Create the portal.

```
apicup subsys create port portal
```

b. For three replica environments, set deployment-profile to `n3xc4.m8`.

```
apicup subsys set port deployment-profile=n3xc4.m8
```

If you omit this step, the deployment-profile defaults to `n1xc4.m8`, which is for non-HA environments, and will not support three instances of the subsystem in one cluster.

c. Specify the license version you purchased.

```
apicup subsys set port
license-use=<license_type>
```

The *license_type* must be either `production` or `nonproduction`. If not specified, the default value is `nonproduction`.

d. Set the portal endpoints.
Endpoints can point to VM host names, but in cluster deployments typically point to a load balancer. The load balancer distributes requests over the 3 VMs. The following values point to a sample load balancer URL.

Component	Command
portal-admin	`apicup subsys set port portal-admin=portal-admin.sample.example.com`
portal-www	`apicup subsys set port portal-www=portal-www.sample.example.com`

e. Set the search domain for the VM.

```
apicup subsys set port search-domain sample.example.com
```

f. Set the DNS Name server for the VM to look up endpoints.

```
apicup subsys set port dns-servers 192.168.1.1
```

g. Set a Public Keyfile.
This is the public key of the user account that you want to use to **ssh** from, to the appliance

```
apicup subsys set port ssh-keyfiles "id_rsa.pub"
```

h. Set a hashed password to access the appliance VM through the VMware Remote Console. Use an operating system utility to create a hashed password, and then use `apicup` to set the hashed password for your subsystem:

```
apicup subsys set port default-password='$1$aTD7uXAO$kNoMAefjGKBwMFiu.8ctr0'
```

Important: Review the requirements for creating and using a hashed password. See Setting and using a hashed default password.

i. Create the hosts for the subsystem. You must specify a password to use to encrypt the disks that the appliance uses.

```
apicup hosts create port portal1.sample.example.com password123
apicup hosts create port portal2.sample.example.com password123
apicup hosts create port portal3.sample.example.com password123
```

Replace the example password in the previous example with a strong password that meets your security requirements.

j. Set the network interface.
Note that the last parameter is the Internet Gateway.

```
apicup iface create port portal1.sample.example.com eth0 192.168.1.107/255.255.255.0 192.168.1.2
apicup iface create port portal2.sample.example.com eth0 192.168.1.108/255.255.255.0 192.168.1.2
apicup iface create port portal3.sample.example.com eth0 192.168.1.109/255.255.255.0 192.168.1.2
```

k. Check the host configuration for problems.

```
apicup hosts list port
```

l. Validate the installation.

```
apicup subsys get port --validate
```

m. Create an ISO file in a plan folder. For example, `portplan-out`.

```
apicup subsys install port --out portplan-out
```

If you have multiple nodes listed for the hosts, when you run the `--out` command, it creates an ISO for each node. When deploying the nodes from VMware, each node gets its own ISO file attached.

n. To deploy the ISOs, see Deploying the Developer Portal subsystem OVA file.

Configuring SSHD to limit access to your deployment

How to configure SSHD in order to limit host access to your VMware deployment, or to change the SSHD cipher level.

About this task

The following instructions show how to configure SSHD in order to limit VMware access to a certain set of IP addresses, and how to change the SSHD cipher level.

Attention:

- SSHD configuration settings are not included in the IBM® API Connect backups. However, the settings are kept when you upgrade to a new fix pack.
- If you change the SSHD cipher level, and encounter issues with the new configuration, you must revert your cipher changes and retest. If after the retest with the original cipher level that came with IBM API Connect you still have issues, contact IBM Support.

Procedure

- The following instructions show you how to configure SSHD in order to limit VMware access to a certain set of IP addresses.
  1. Log in to the target virtual machine by using an SSH tool, and switch to the root user. For example, to log in to the Management subsystem, run the following commands:

     ```
     ssh ip_address -l apicadm
     sudo -i
     ```

  2. Move to the following folder:

     ```
     /etc/ssh/sshd_config.d
     ```

  3. Create a new configuration file with the extension .conf that contains the IP address of the host that you want to allow to access your deployment. For example, to create the file run the following command:

     ```
     vi filename.conf
     ```

     Then, edit the file by using the following format, which in this example is allowing users from IP address `123.135.1.2`:

     ```
     AllowUsers *@123.135.1.2
     ```

     Additional hosts can be added by separating the IP address patterns with a space. For example:

     ```
     AllowUsers *@123.135.1.2 *@135.168.1.2
     ```

  4. Restart the SSHD service by running the following command:

     ```
     systemctl restart sshd.service
     ```

- The following instructions show you how to configure SSHD to change the cipher level.
  1. Log in to the target virtual machine by using an SSH tool, and switch to the root user. For example, to log in to the Management subsystem, run the following commands:

     ```
     ssh ip_address -l apicadm
     sudo -i
     ```

  2. Move to the following folder:

     ```
     /etc/ssh/sshd_config.d
     ```

  3. Create a new configuration file with the extension .conf that contains the list of ciphers that you want to support. For example, if you want to remove the 128 bit UMAC cipher from the following list of currently supported ciphers:

     ```
     MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
     ```

     Create the .conf file by running the following command:

     ```
     vi filename.conf
     ```

     Then, remove that cipher from the list by adding the following line to the file:

     ```
     MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512,hmac-sha2-256
     ```

  4. Restart the SSHD service by running the following command:

     ```
     systemctl restart sshd.service
     ```

Results

Your SSHD configuration is updated to limit access to your VMware to a certain set of IP addresses, or to change the cipher level.

Enabling Developer Portal feature flags on VMware

Use a template override to update the installation CR with settings that control the default behavior of the Developer Portal subsystem.

About this task

You can add a template override to the Developer Portal subsystem CR to change the default behavior of the Portal subsystem, or to turn specific features on and off. Table 1 describes the available feature flags.

Table 1. List of available feature flags for the Developer Portal

Feature flag name	Default Value	Pod	Container	Description

`PORTAL_SKIP_WEB_ENDPOINT_VALIDATION`	`true`	`www`	`admin`	Controls whether the Portal subsystem checks whether the Portal web endpoint is accessible.
				
				Sometimes, the endpoint cannot be reached, for example due to the complexity of the networks. In which case the following type of error can be seen in the `portal-www` pod, and `admin` container logs:
				
				`An error occurred contacting the provided portal web endpoint: example.com`
				`The provided Portal web endpoint example.com returned HTTP status code 504`
				
				You can disable the check by setting this flag to `false`.

`ENABLE_TRUSTED_REVERSE_PROXY`	`false`	`www`	`admin`	Controls whether the Drupal `trusted_reverse_proxy` module can manage the reverse proxies list.
				
				If you have multiple load balancers and reverse proxies in your upstream network, and the perimeter module is triggered, the incorrect IP address might be banned rather than the intended client IP address. However, you can manage which IP addresses are banned by using the Drupal `trusted_reverse_proxy` module, and setting this feature flag to `true`.
				
				The Drupal module inspects the `x-forwarded-for` headers to identify the reverse proxies, and trusts that the first IP address found in the list is the client IP. For example, in `X-Forwarded-For: <client>, <proxy1>, <proxy2>`, the client IP `<client>` will be banned. For more information about this module, see https://www.drupal.org/project/trusted_reverse_proxy.
				
				To use this module, you must ensure that all of the proxies in your upstream network, including the ingress controller, are configured correctly so that the `x-forwarded-for` header is passed through to the `portal-www` pod. For example, the default behavior of the `nginx-ingress-controller` is to ignore the inbound `x-forwarded-for` header, and create a new one. To change this behavior, update the `nginx-ingress-controller` `configmap` with the following information:
				
				`data:`
				`  compute-full-forwarded-for: "true"`
				`  use-forwarded-headers: "true"`
				
				Warning: Enable this feature flag only if you trust your upstream reverse proxies, as it is possible to trick the `x-forwarded-for` header.

Procedure

1. Create an `extra-values.yaml` file, or open an existing `extra-values.yaml` file.
2. In the file, append the feature flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.
   The following example enables the `trusted_reverse_proxy` feature on all possible UI pages.

   ```
   spec:
     template:
     - containers:
       - env:
          - name: ENABLE_TRUSTED_REVERSE_PROXY
            value: "true"
          name: container
       name: pod
   ```

   Where:
   - *container* is the name of the portal container.
   - *pod* is the name of the portal pod.

   Note: You can add multiple feature flags to the file.
3. Save the file.
4. Run the following command to set the `extra-values.yaml` file:

   ```
   apicup subsys set <portal_subsystem> extra-values-file <path_to_portal_extra-values.yaml_file>
   ```

5. Apply the overrides to the portal subsystem by running the following command:

   ```
   apicup subsys install <SUBSYSTEM_NAME>
   ```

   The Developer Portal subsystem is updated to enable or disable features as specified.

Managing an appliance data disk

You can use **apic** commands to manage appliance data disks in your VMware deployment.

The API Connect deployment on VMware uses 2 partitions. The first contains the base operating system, while the second is encrypted so that any customer data that is stored on disk uses the encrypted volume. Encryption is done with Linux Unified Key Setup (LUKS) disk encryption.

During installation of each API Connect subsystem, you specify a password **HD-PASSWD** when configuring the host:

```
apicup hosts create <SUBSYS> <HOSTNAME> <HD-PASSWD>
```

If you want to force a restart of processes, without requiring a full restart of the virtual machine, you can use the command **apic lock** to stop the Kubernetes node on the virtual machine and lock the secured storage. When you are ready to restart processes, you can use **apic unlock** to restart the Kubernetes node. The command **apic unlock** uses the password to unlock the partition so that files can be read from it and written to it. The unlock command also starts the **apic** daemon, also known as the **appliance-manager** service.

You can check that status of this service with:

```
sudo systemctl status appliance-manager
```

One **appliance-manager** daemon runs on each node, where it manages the Kubernetes cluster, processes update and upgrade requests, and gathers logs.

apic command	Description
lock	Shut down appliance services and lock the secured storage
logs	Retrieve logs from all nodes in the cluster
status	Report on cluster status
unlock	Unlock the secure storage and start the appliance
version	Get the API Connect appliance base version

Replica override for microservices on VMware

Modify the replica count for microservices by using template overrides for Custom Resources.

When API Connect is deployed, the deployment profile you select drives the number of replicas to be used for most deployment. For one replica profiles, specified by **n1x...**, the replica count is 1. For three replica profiles, specified by **n3x...**, the replica count is 3.

You can override the replica count for individual deployments by using template overrides.

- Template overrides are a piece of yaml that should be added to the Kubernetes custom resource. The examples provided in this topic show only that the **template:** element should be under the main **spec:** element of a CR. The rest of the CR is not shown.
- When applying overrides for Appliance deployments (virtual machines), the yaml sample provided as example must be added in an extra-values-file for the subsystem. To view an example of documenting a template override for an Appliance, see Setting rate limits for public APIs on the management service

Management subsystem

For a Management CR in Kubernetes or OpenShift, the following override specifies 5 replicas for the **apim** deployment:

```
spec:
  template:
  - name: apim
    replicaCount: 5
```

You can also configure replicate count through a similar template override for:

- **juhu**
- **ldap**
- **lur**

Analytics subsystem

The Analytics microservices that can be scaled using template override are:

- **ingestion**
- **mtls-gw**
- **storage-shared**
  Note that this microservice cannot be scaled below 2 replicas when using a three replica deployment profile.

Testing a new configuration against a current cluster

You can use the **apicup** utility to test the validity of a new configuration before you install it.

The **apicup** utility is used to set configuration properties for subsystems on API Connect on VMware appliances. The utility can be used for installation, upgrade, and maintenance activities such as backup and restore.

The usage pattern is to set values for each parameter you need for a subsystem, and then deploy the new values with the **apicup subsys install** command.

You can use the **--dry-run** option to test the validity of your values prior to deploying them to an existing cluster:

```
apicup subsys install <subsystem_name> --dry-run
```

Important: You cannot use **apicup subsys install <SUBSYS> --dry-run** for a fresh install or an upgrade. The **--dry-run** feature is for testing proposed configuration changes on an existing cluster.
For example, if you want to configure S3 backups for you management subsystem, the command flow is:

1. Follow the instructions for configuring S3 backupsReconfiguring backup settings for the management subsystem V10.0.6.0 by issuing the required **apicup subsys set** commands:

   ```
   apicup subsys set mgmt <setting>=<value>
   apicup subsys set mgmt <second_setting>=<value>
   apicup subsys set mgmt <third_setting>=<value>
   .
   .
   .
   ```

2. Do a dry run of the install command:

   ```
   apicup subsys install mgmt --dry-run
   ```

   Example message on success:

   ```
   managementcluster.management.apiconnect.ibm.com/management configured (server dry run)
   ```

   Example message on failure:

   ```
   Error: Failed to execute dry-run:....
   ```

   The error message also contains details about the error.

3. When dry run returns success, deploy:

   ```
   apicup subsys install mgmt
   ```

TLS certificate concepts and operations

Information about internal API Connect certificates, renewing expired certificates, exporting certificates, and other certificate maintenance operations.

This topic covers advanced certificate configuration procedures that are not required in day-to-day API Connect operations. For best practices relating to API Connect certificates, see API Connect TLS certificate best practices.

API Connect certificate types

All TLS certificates that are generated and used by API Connect fall into one of the following categories:

- CA certificates - API Connect creates self-signed CA certificates to sign all the API Connect end-entity certificates. The CA certificates are:
  - **ingress-ca** - used by the ingress-issuer. The ingress-ca signs all user-facing and inter-subsystem certificates.
  - **root-ca** - a self-signed root certificate that signs the **ingress-ca** certificate.
  - **management-ca** - signs all the management intra-subsystem certificates.
  - **portal-ca** - signs all the portal intra-subsystem certificates.
  - **analytics-ca** - signs all the analytics intra-subsystem certificates.

  CA certificates have a default duration of 10 years.
  Do not customize CA certificates. The only scenarios when CA certificates require updating are:
  - Expiry, after 10 years.
  - Synchronizing the **ingress-ca** certificate between data centers in multiple data center deployments.
- Ingress certificates - Ingress certificates are end-entity certificates that are signed by the **ingress-ca** certificate and secure the API Connect endpoints. Ingress certificates are divided into two types:
  - Inter-subsystem certificates - certificates that are used to secure the communication between API Connect subsystems, for example between the management subsystem and gateway. By default, communication between most subsystems uses mTLS. It is possible to use JWT instead of mTLS: Using JWT instead of mTLS on OVA.
    Do not customize inter-subsystem certificates.
  - User-facing certificates - certificates that API Connect users see when they interact with the API Connect UIs and REST API.
    You can customize your user-facing certificates: Customizing your user-facing certificates

  All the ingress certificates have a duration of 2 years.
- Intra-subsystem certificates - certificates that are used between software components in the same subsystem. Intra-subsystem certificates are signed by a subsystem specific CA certificate: **management-ca**, **portal-ca**, **analytics-ca**. The gateway subsystem is a single pod, and has no intra-subsystem certificates.
  Do not customize intra-subsystem certificates.

Note: The inter and intra-subsystem certificates use of a self-signed root CA presents no security risk. For internal communications with a limited number of predefined endpoints, use of locally generated self-signed CA certificates can be more secure than external CA certificates, as certificates are only ever issued for the known components and the domain of trust is kept as small as possible.

- **Certificate reference VMware**
  Details of all API Connect TLS certificates on VMware.
- **Customizing user-facing certificates**
  How to set the user-facing endpoints of API Connect to use your own TLS server certificates.
- **Certificate expiry and renewal**
  How to renew and replace TLS certificates in your VMware deployment.
- **Extracting and copying certificates**
  How to extract certificate contents and apply them to API Connect subsystems in other environments.
- **Rotating the portal intra-subsystem certificates on VMware**
  Use the **apicup** command to rotate the portal intra-subsystem certificates.
- **Apicup certs command reference**
  Documentation of the **apicup certs** command you use to manage API Connect TLS certificates and secrets.
- **Certificate validation reference**
  Certificates validation errors reference table.

Certificate reference VMware

Details of all API Connect TLS certificates on VMware.

The **apicup certs list** command returns all the API Connect TLS certificates that can be configured with the **apicup** command.
Note: The **apicup certs list** command also returns credentials and encryption secrets that are used by API Connect. For information about these secrets and credentials, see Credentials and encryption secrets reference.

Common certificates

Common certificates exist in all subsystems of an API Connect deployment, and must be identical in each subsystem to enable them to communicate with each other.

Because you deploy your API Connect subsystems from a single project directory, these certificates are automatically set to be the same on all subsystems.

Do not customize any of the common certificates.

Table 1. Common certificates

Certificate name	Description
`root-ca`	CA certificate that is the root of the certificate chain for all other API Connect certificates. Do not update this certificate.
`ingress-ca`	Intermediate CA certificate used to generate all API Connect ingress certificates. Do not customize this certificate. If the certificate expires (duration is 20 years), see: Renewing ingress-ca. If you have multiple data centers that are managed from different project directories, you can synchronize this certificate, see: Synchronizing the ingress-ca certificate.
`appliance-client`	A client certificate that is used by the **apicup** command to communicate with the API Connect subsystem appliances. Requires EKU `clientAuth`. Do not update this certificate.
`k8s-ca`	CA certificate that forms the root of the certificate chain for the Kubernetes cluster components that run in the API Connect subsystem appliance. Do not update this certificate.

Management certificates

The following table lists the certificates that exist on the management subsystem, in addition to the common certificates.

Table 2. Management subsystem certificates

Certificate name	Description
`portal-client`	The `portal-client` certificate is an inter-subsystem ingress certificate that is presented to the developer portal subsystem during portal service registration. For successful portal registration the `portal-client` certificate must meet these requirements: • Specify EKU `clientAuth`. • Signed by the same CA (`ingress-ca`) as the `portal-admin-ingress` certificate that secures the admin endpoint on the developer portal. • The subject name must match the `adminClientSubjectDN` in the portal subsystem. If you have a multiple data center deployment, both data centers must use an identical subject name. Do not update this certificate.
`analytics-client-client`	A legacy certificate, it is not used from v10.0.5 onwards.

Certificate name	Description
`analytics-ingestion-client`	The `analytics-ingestion-client` certificate is an inter-subsystem ingress certificate that is presented to the analytics subsystem during analytics service registration. For successful analytics registration the `analytics-ingestion-client` certificate must meet these requirements: • Specify EKU `clientAuth`. • Signed by the same CA (`ingress-ca`) as the `analytics-ingestion-ingress` certificate that secures the ingestion endpoint on the analytics subsystem. • The subject name must match the `clientSubjectDN` in the analytics subsystem. `analytics-ingestion-client` is also used by the gateway subsystem to authenticate with the analytics ingestion endpoint, for the transmission of API event data. For multiple data center deployments, both data centers must have an identical subject name. Do not update this certificate.
`platform-api`	The `platform-api` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at cloud scope. The endpoint in the management subsystem configuration uses the same name: `platform-api` The hostnames for which the certificate is valid must include the `platform-api` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`consumer-api`	The `consumer-api` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at organization scope. The endpoint in the management subsystem configuration uses the same name: `consumer-api` The hostnames for which the certificate is valid must include the `consumer-api` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`api-manager-ui`	The `api-manager-ui` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Manager UI. The endpoint in the management subsystem configuration uses the same name: `api-manager-ui`. The hostnames for which the certificate is valid must include the `api-manager-ui` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`cloud-admin-ui`	The `cloud-admin-ui` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the Cloud Manager UI. The endpoint in the management subsystem configuration uses the same name: `cloud-admin-ui`. The hostnames for which the certificate is valid must include the `cloud-admin-ui` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`hub`	Used by the Automated API behavior testing application. User-facing ingress certificate for the `hub-endpoint`.
`turnstile`	Used by the Automated API behavior testing application. User-facing ingress certificate for the `turnstile-endpoint`.
`management-replication-client`	Used in 2DCDR deployments only. The `management-replication-client` certificate is an inter-subsystem ingress client certificate that is used when connecting to the `management-replication` endpoint in the other data center. Requires EKU `clientAuth`.
`management-replication-ingress`	Used in 2DCDR deployments only. The `management-replication-ingress` certificate is an inter-subsystem ingress server certificate that is used to secure the `management-replication` endpoint. `management-replication-ingress` must be valid for the endpoint defined by `apicup subsys set <subsystem> management-replication=<endpoint>` Requires EKU `serverAuth`.

Portal subsystem certificates

The following table lists the certificates that exist on the management subsystem, in addition to the common certificates.

Table 3. Portal subsystem certificates

Certificate name	Description

Certificate name	Description
`mgmt-consumer-api`	The `mgmt-consumer-api` certificate is an inter-subsystem ingress certificate that secures the consumer API endpoint on the management subsystem. The portal has a copy of this certificate so it can verify that it is communicating with the management subsystem. This certificate must be identical to the `consumer-api` certificate in the management subsystem. Do not update this certificate.
`mgmt-platform-api`	The `mgmt-platform-api` certificate is an inter-subsystem ingress certificate that secures the platform API endpoint on the management subsystem. The portal has a copy of this certificate so it can verify that it is communicating with the management subsystem. This certificate must be identical to the `platform-api` certificate in the management subsystem. Do not update this certificate.
`portal-admin-ingress`	The `portal-admin-ingress` certificate is an inter-subsystem ingress certificate that is used to secure the portal administration endpoint. The management subsystem communicates with this endpoint to register the portal and to send portal configuration updates. Do not update this certificate.
`portal-replication-client`	Used in 2DCDR deployments only. The `portal-replication-client` certificate is an inter-subsystem ingress client certificate that is used when connecting to the `portal-replication` endpoint in the other data center. Requires EKU `clientAuth`.
`portal-replication-ingress`	Used in 2DCDR deployments only. The `portal-replication-ingress` certificate is an inter-subsystem ingress server certificate that is used to secure the `portal-replication` endpoint. `portal-replication-ingress` must be valid for the endpoint defined by `apicup subsys set <subsystem> portal-replication=<endpoint>` Requires EKU `serverAuth`.
`portal-www-ingress`	The `portal-www-ingress` certificate is a user-facing ingress certificate that is returned to users when they attempt to access developer portal sites. The endpoint is called `portal-www` in the portal subsystem configuration.

Analytics subsystem certificates

The following table lists the certificates that exist on the management subsystem, in addition to the common certificates.

Table 4. Analytics subsystem certificates

Certificate name	Description
`analytics-ca`	Internal certificate used for communication between analytics subsystem components. Do not update this certificate.
`analytics-ingestion-ingress`	The `analytics-ingestion-ingress` certificate is an inter-subsystem ingress certificate that is used to secure the analytics ingestion endpoint, which is used for the following: • Receiving API event data from associated gateways. • Configuration and analytics queries from the management subsystem. Do not update this certificate.
`service-client`	Internal certificate used for communication between analytics subsystem components. Do not update this certificate.
`service-server`	Internal certificate used for communication between analytics subsystem components. Do not update this certificate.

Ingress certificates

The following table lists all the certificates across all subsystems that are signed by the `ingress-ca`. If you renew your `ingress-ca` certificate, then you must also renew all the ingress end-entity certificates that are listed in this table. For steps on renewing the `ingress-ca`, see Renewing ingress-ca.

Table 5. Certificates signed by `ingress-ca`

Certificate name	Description
`portal-client`	The `portal-client` certificate is an inter-subsystem ingress certificate that is presented to the developer portal subsystem during portal service registration. For successful portal registration the `portal-client` certificate must meet these requirements: • Specify EKU `clientAuth`. • Signed by the same CA (`ingress-ca`) as the `portal-admin-ingress` certificate that secures the admin endpoint on the developer portal. • The subject name must match the `adminClientSubjectDN` in the portal subsystem. If you have a multiple data center deployment, both data centers must use an identical subject name. Do not update this certificate.
`analytics-client-client`	A legacy certificate, it is not used from v10.0.5 onwards.

Certificate name	Description
`analytics-ingestion-client`	The `analytics-ingestion-client` certificate is an inter-subsystem ingress certificate that is presented to the analytics subsystem during analytics service registration. For successful analytics registration the `analytics-ingestion-client` certificate must meet these requirements: • Specify EKU `clientAuth`. • Signed by the same CA (`ingress-ca`) as the `analytics-ingestion-ingress` certificate that secures the ingestion endpoint on the analytics subsystem. • The subject name must match the `clientSubjectDN` in the analytics subsystem. `analytics-ingestion-client` is also used by the gateway subsystem to authenticate with the analytics ingestion endpoint, for the transmission of API event data. For multiple data center deployments, both data centers must have an identical subject name. Do not update this certificate.
`platform-api`	The `platform-api` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at cloud scope. The endpoint in the management subsystem configuration uses the same name: `platform-api` The hostnames for which the certificate is valid must include the `platform-api` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`consumer-api`	The `consumer-api` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at organization scope. The endpoint in the management subsystem configuration uses the same name: `consumer-api` The hostnames for which the certificate is valid must include the `consumer-api` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`api-manager-ui`	The `api-manager-ui` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Manager UI. The endpoint in the management subsystem configuration uses the same name: `api-manager-ui`. The hostnames for which the certificate is valid must include the `api-manager-ui` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`cloud-admin-ui`	The `cloud-admin-ui` certificate is a user-facing ingress certificate that is returned to users when they attempt to access the Cloud Manager UI. The endpoint in the management subsystem configuration uses the same name: `cloud-admin-ui`. The hostnames for which the certificate is valid must include the `cloud-admin-ui` endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU `serverAuth`.
`hub`	Used by the Automated API behavior testing application. User-facing ingress certificate for the `hub-endpoint`.
`turnstile`	Used by the Automated API behavior testing application. User-facing ingress certificate for the `turnstile-endpoint`.
`portal-admin-ingress`	The `portal-admin-ingress` certificate is an inter-subsystem ingress certificate that is used to secure the portal administration endpoint. The management subsystem communicates with this endpoint to register the portal and to send portal configuration updates. Do not update this certificate.
`portal-www-ingress`	The `portal-www-ingress` certificate is a user-facing ingress certificate that is returned to users when they attempt to access developer portal sites. The endpoint is called `portal-www` in the portal subsystem configuration.
`analytics-ingestion-ingress`	The `analytics-ingestion-ingress` certificate is an inter-subsystem ingress certificate that is used to secure the analytics ingestion endpoint, which is used for the following: • Receiving API event data from associated gateways. • Configuration and analytics queries from the management subsystem. Do not update this certificate.
`management-replication-client`	Used in 2DCDR deployments only. The `management-replication-client` certificate is an inter-subsystem ingress client certificate that is used when connecting to the `management-replication` endpoint in the other data center. Requires EKU `clientAuth`.

Certificate name	Description
`management-replication-ingress`	Used in 2DCDR deployments only. The **`management-replication-ingress`** certificate is an inter-subsystem ingress server certificate that is used to secure the **`management-replication`** endpoint. **`management-replication-ingress`** must be valid for the endpoint defined by **`apicup subsys set <subsystem> management-replication=<endpoint>`** Requires EKU **`serverAuth`**.
`mgmt-consumer-api`	The **`mgmt-consumer-api`** certificate is an inter-subsystem ingress certificate that secures the consumer API endpoint on the management subsystem. The portal has a copy of this certificate so it can verify that it is communicating with the management subsystem. This certificate must be identical to the **`consumer-api`** certificate in the management subsystem. Do not update this certificate.
`mgmt-platform-api`	The **`mgmt-platform-api`** certificate is an inter-subsystem ingress certificate that secures the platform API endpoint on the management subsystem. The portal has a copy of this certificate so it can verify that it is communicating with the management subsystem. This certificate must be identical to the **`platform-api`** certificate in the management subsystem. Do not update this certificate.
`portal-replication-client`	Used in 2DCDR deployments only. The **`portal-replication-client`** certificate is an inter-subsystem ingress client certificate that is used when connecting to the **`portal-replication`** endpoint in the other data center. Requires EKU **`clientAuth`**.
`portal-replication-ingress`	Used in 2DCDR deployments only. The **`portal-replication-ingress`** certificate is an inter-subsystem ingress server certificate that is used to secure the **`portal-replication`** endpoint. **`portal-replication-ingress`** must be valid for the endpoint defined by **`apicup subsys set <subsystem> portal-replication=<endpoint>`** Requires EKU **`serverAuth`**.

User-facing certificates

The following table lists all the certificates that are used for user-facing endpoints.

Table 6. User-facing endpoint certificates.

Certificate name	Description
`portal-www-ingress`	The **`portal-www-ingress`** certificate is a user-facing ingress certificate that is returned to users when they attempt to access developer portal sites. The endpoint is called **`portal-www`** in the portal subsystem configuration.
`api-manager-ui`	The **`api-manager-ui`** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Manager UI. The endpoint in the management subsystem configuration uses the same name: **`api-manager-ui`**. The hostnames for which the certificate is valid must include the **`api-manager-ui`** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **`serverAuth`**.
`cloud-admin-ui`	The **`cloud-admin-ui`** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the Cloud Manager UI. The endpoint in the management subsystem configuration uses the same name: **`cloud-admin-ui`**. The hostnames for which the certificate is valid must include the **`cloud-admin-ui`** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **`serverAuth`**.
`consumer-api`	The **`consumer-api`** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at organization scope. The endpoint in the management subsystem configuration uses the same name: **`consumer-api`** The hostnames for which the certificate is valid must include the **`consumer-api`** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **`serverAuth`**.
`platform-api`	The **`platform-api`** certificate is a user-facing ingress certificate that is returned to users when they attempt to access the API Connect REST API at cloud scope. The endpoint in the management subsystem configuration uses the same name: **`platform-api`** The hostnames for which the certificate is valid must include the **`platform-api`** endpoint. A wildcard certificate can be used as the first element in a hostname, for example, if the endpoint is: *store.acme.com*, the certificate can be one that is valid for hostnames that match **.acme.com*. Requires EKU **`serverAuth`**.
`hub`	Used by the Automated API behavior testing application. User-facing ingress certificate for the **`hub-endpoint`**.

Certificate name	Description
`turnstile`	Used by the Automated API behavior testing application. User-facing ingress certificate for the `turnstile-endpoint`.

Customizing user-facing certificates

How to set the user-facing endpoints of API Connect to use your own TLS server certificates.

Before you begin

Identify the user-facing certificates that you want to customize: Table 6.

Ensure that you have certificate, key, and CA PEM files for each certificate you want to customize. The PEM files must not be password-protected.

For example, to customize the certificate that secures the Cloud Manager UI, you must have the following PEM files: cloud-admin-ui-cert.pem, cloud-admin-ui-key.pem, cloud-admin-ui-ca.pem

User-facing certificates must meet the following requirements:

- Specify Extended Key Usage (EKU) of `serverAuth`.
- Include the Subject Alternative Name (SAN) of the endpoint that they secure.
- If the certificate is signed by an intermediate CA, then the ca.pem file must contain the full chain to root CA.

About this task

You update the user-facing certificates with the **apicup** command.

You can set custom user-facing certificates during installation, or you can update them after installation, replacing the default certificates that are generated.

Note: API Connect verifies certificates when you set them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your custom certificates before they expire.

Procedure

1. If you are updating a user-facing certificate after installation, you must first clear the default certificate that was created:

   ```
   apicup certs set <subsystem name> <certificate name> --clear
   ```

   For example, to clear the certificate that secures the Cloud Manager UI:

   ```
   apicup certs set mgmt cloud-admin-ui --clear
   ```

2. Set the user-facing certificate with the **apicup certs set** command, supplying the PEM files for your new certificate:

   ```
   apicup certs set <subsystem name> <certificate name> <cert file> <key_file> <ca file>
   ```

   For example:

   ```
   apicup certs set mgmt cloud-admin-ui cert.pem key.pem ca.pem
   ```

3. If you are updating the certificate after installation, then apply the update with `apicup subsys install`

   ```
   apicup subsys install <subsystem name>
   ```

4. You can verify that the certificate is updated with **apicup certs list**:

   ```
   apicup certs list <subsystem name>
   ```

   Example output:

   ```
   Common certificates
   ===================

   Name                        Summary                                                                 Validation errors
   ----                        -------                                                                 -----------------
   ...

   Subsystem def-management certificates
   =====================================

   Name                              Summary                                                           Validation errors
   ----                              -------                                                           -----------------
   ...
   cloud-admin-ui                    CN: cloud-admin-ui
                                     SubjectKeyId: 9E:8B:FA:7E:92:70:EC:86:2F:C3:3A:20:E8:13:EA:F4
                                     AuthorityKeyId: E4:AB:09:FF:A9:7D:D3:70:DD:AD:69:C4:A4:34:8D:96
   ...
   ```

   Note: Along with the user-facing certificates, the **apicup certs list** command outputs details of other internal API Connect certificates and secrets that you should not update.

Certificate expiry and renewal

How to renew and replace TLS certificates in your VMware deployment.

The instructions in this topic apply only to the management, portal, and analytics subsystems. For gateway appliance-specific instructions, see *Configuring the API Connect Gateway Service* in the appropriate version of the DataPower documentation.
Note: The API invocation certificate (the certificate that is presented to callers of your APIs) is configured in the Cloud Manager UI when you register your gateway services. For more information about API invocation certificates, see Registering a gateway service and TLS profiles overview.
All ingress certificates in API Connect have a duration of 2 years. To avoid an outage when they expire, they must be manually renewed before expiration.

The **ingress-ca** CA certificate has a duration of 10 years and also requires manual steps to renew it before expiry.

Intra-subsystem certificates are renewed automatically.

Identifying expired certificates

The **apicup** tool does not check certificate expiry, to check for expired certificates you must login to your VMs and use the **apicops** tool:

1. Login to each subsystem VM:

   ```
   ssh apicadm@<subsystem hostname>
   ```

   Note: If you have a three replica deployment, login to only one of the VMs of each subsystem.
2. Switch to the root user:

   ```
   sudo -i
   ```

3. Install the **apicops** tool: The API Connect operations tool: apicops:
   a. Download the latest **apicops-v10-linux** executable file:

      ```
      wget <URL of latest apicops-v10-linux>
      ```

      For example:

      ```
      wget https://github.com/ibm-apiconnect/apicops/releases/download/<latest version>/apicops-v10-linux
      ```

   b. Add execute permissions to **apicops-v10-linux**:

      ```
      chmod a+x apicops-v10-linux
      ```

4. Run the **apicops certs:validate** command:

   ```
   ./apicops-v10-linux certs:validate -s <subsystem>
   ```

   where *<subsystem>* is the subsystem you are running the command on, either: Management, portal, or analytics.
   The **apicops** command reports any certificates that are expired or within 60 days of expiry.

   You can ignore the message:

   ```
   Cannot check who signed the certificate returned by ... as no ingress issuer was provided
   ```

   this is expected on VMware deployments.
5. Renew any certificates that are expired or approaching expiry, except for the certificates that are listed under the heading **Validating subsystem certificates for <subsystem name>**, these are intra-subsystem certificates that are renewed automatically.

   ```
   Validating subsystem certificates for abc-management
   Good certificate: <subsystem name>-ca
   Good certificate: <subsystem name>-client
   Good certificate: <subsystem name>-server
   ```

   Follow the step in Renewing certificates to renew expired or nearly expired certificates.

Renewing certificates

Use the **apicup** command to renew certificates.
Warning: Use these steps to renew only end-entity ingress certificates as defined here: ingress certificates. If you want to renew the **ingress-ca** certificate, then see Renewing ingress-ca.

1. Clear the existing certificate with the **apicup certs clear** command:

   ```
   apicup certs set --clear <subsystem> <certificate name>
   ```

   Repeat for all ingress certificates that you want to renew.
2. Generate new certificates for all the certificates that you cleared in step 1:

   ```
   apicup certs generate <subsystem>
   ```

   You do not need to specify the certificates with the **apicup certs generate**, the command identifies all empty (cleared) certificates in the subsystem and generates new ones.
3. Apply the updated certificates to the subsystem:

   ```
   apicup subsys install <subsystem>
   ```

Renewing `ingress-ca`

Use the **apicup** command to renew **ingress-ca**, and then renew all the end-entity certificates the are signed by **ingress-ca**.

1. Clear the **ingress-ca** certificate:

```
apicup certs set --clear <management subsystem> ingress-ca
```

Although this command specifies *<management subsystem>*, because **ingress-ca** is a common certificate, it is cleared for all subsystems. You do not need to repeat this command for every portal or analytics subsystem that is managed from the same project directory.

2. Generate a new **ingress-ca**:

```
apicup certs generate <subsystem>
```

The **generate** command creates certificates for all previously cleared certificates, you do not need to specify **ingress-ca** in this command.

3. Clear all the ingress end-entity certificates that are signed by **ingress-ca**:

```
apicup certs set --clear <subsystem> <certificate name>
```

For a list of the ingress certificates, see Ingress certificates.

4. Generate new ingress end-entity certificates for each subsystem:

```
apicup certs generate <subsystem>
```

5. Apply the updated certificates to all subsystems:

```
apicup subsystem install <subsystem>
```

Extracting and copying certificates

How to extract certificate contents and apply them to API Connect subsystems in other environments.

If you have a two data center disaster recovery deployment, or have API Connect subsystems in different data centers that are managed by different project directories, then it is necessary to manually synchronize the TLS certificates between the environments.

Synchronizing the `ingress-ca` certificate

To export the **ingress-ca** certificate from your target environment, follow these steps:

1. Extract the **ingress-ca** public certificate to a file called ingress-ca-cert.pem

```
apicup certs get <subsystem> ingress-ca > ingress-ca-cert.pem
```

2. Extract the **ingress-ca** private key to a file called ingress-ca-key.pem

```
apicup certs get <subsystem> ingress-ca -t key > ingress-ca-key.pem
```

3. Extract the **ingress-ca** CA to a file called ingress-ca-CA.pem

```
apicup certs get <subsystem> ingress-ca -t ca > ingress-ca-CA.pem
```

Note: The **ingress-ca** certificate is the same for all subsystems that are managed from the same project directory, so you can specify any subsystem in the **apicup certs get** command.
To apply the exported **ingress-ca** certificate to another environment, follow these steps:

1. Apply the exported certificate files to the **ingress-ca**:

```
apicup certs set <subsystem> ingress-ca ingress-ca-cert.pem ingress-ca-key.pem ingress-ca-CA.pem
```

Note: The **ingress-ca** certificate is the same for all subsystems that are managed from the same project directory, so you can specify any subsystem in the **apicup certs set** command.

2. Clear all the ingress end-entity certificates that are signed by **ingress-ca**:

```
apicup certs set --clear <subsystem> <certificate name>
```

For a list of the ingress certificates, see Ingress certificates.

3. Generate new ingress end-entity certificates for each subsystem:

```
apicup certs generate <subsystem>
```

4. Apply the updated certificates to all subsystems:

```
apicup subsystem install <subsystem>
```

Rotating the portal intra-subsystem certificates on VMware

Use the **apicup** command to rotate the portal intra-subsystem certificates.

The portal's intra-subsystem certificates: **portal-ca**, **portal-client**, and **portal-server**, are not shown in the **apicup certs list** output, and cannot be updated with **apicup certs set**. If you want to renew these certificates, you can use the **apicup subsys rotate-secrets** command.
Warning: The portal intra-subsystem certificates are internal certificates that are created and renewed automatically. Renewing these certificates with **apicup subsys rotate-secrets** is only necessary in specific failure scenarios and should be done only if advised by IBM support.
To rotate the Developer Portal subsystem secrets, certificates, or both, run the following command:

```
apicup subsys rotate-secrets portal
```

```
Rotate portal subsystem secrets

Usage:
  apicup subsys rotate-secrets SUBSYS [flags]

Flags:
      --certs strings              List of certificates you want to rotate (Listing an issuer will rotate any certs issued by
the issuer e.g listing just portal-ca will rotate portal-ca, portal-client and portal-server)
      --encryption-secret string   Optional value to set the encryption secret to, if supplied secret exists in the env. If not
supplied the rotated secret will be random.
  -h, --help                       help for rotate-secrets
      --rotate-encryption-secret   Set to true if you want to rotate the encryption secret.
      --wait                       Wait for the operation to complete or fail.
      --wait-timeout duration      Command timeout in seconds. (default 40s)

Global Flags:
      --accept-license   Deprecated. Use apicup licenses commands to work with licenses
      --debug            Enable debug logging
```

For example, to rotate the encryption secret to a random value run the following command:

```
apicup subsys rotate-secrets portal --rotate-encryption-secret true
```

The following example rotates the encryption secret to a named secret (note that first you must exec into the Developer Portal OVA to create the secret):

```
apicup subsys rotate-secrets portal --rotate-encryption-secret --encryption-secret my-new-secret
```

The following example rotates all of the Developer Portal certificates:

```
apicup subsys rotate-secrets portal --certs portal-ca
```

The following example rotates just the server and client certificates:

```
apicup subsys rotate-secrets portal --certs portal-server,portal-client
```

To list any Developer Portal secret rotations that were created by using the `rotate-secrets` command:

```
apicup subsys list-rotate-secrets portal --timeout 30
```

Where:

- *portal* is the name of your Developer Portal subsystem.
- `--timeout` (optional) is the duration in seconds of the command timeout. Default is 40 seconds.

To delete any Developer Portal secret rotations that were created by using the `rotate-secrets` command:

```
apicup subsys delete-rotate-secret portal --rotate-secret-name secret_name --wait true --wait-timeout 80

Delete a secret rotation of the subsystem

Usage:
  apicup subsys delete-rotate-secret SUBSYS [flags]

Flags:
  -h, --help                         help for delete-rotate-secret
      --rotate-secret-name string    Value of a Portal Secret Rotation CR name to delete.
      --wait                         Wait for the operation to complete or fail.
      --wait-timeout duration        Command timeout in seconds. (default 1m0s)

Global Flags:
      --accept-license   Deprecated. Use apicup licenses commands to work with licenses
      --debug            Enable debug logging
```

Apicup certs command reference

Documentation of the **apicup certs** command you use to manage API Connect TLS certificates and secrets.

The **apicup certs** command is used to set, clear, and generate the TLS certificates that are used by API Connect. The **apicup certs** command is also used to manage some of the secrets used by API Connect, such as remote backup server credentials and database encryption secrets.

After you modify certificates with the **apicup certs** command, you must apply the changes to your subsystems with:

```
apicup subsys install <subsystem name>
```

Setting and clearing certificates

Run the `apicup certs set` command:

Table 1. apicup certs set

Command	Values	Result

Command	Values	Result
`apicup certs set SUBSYS CERT_NAME [CERT_FILE KEY_FILE CA_FILE] [flags]`	Parameters are: • SUBSYS - name of the subsystem to which the certificate applies • CERT_NAME - name of the certificate; see Certificate reference VMware for a list of certificates that can be set for each subsystem. • CERT_FILE - Path to the certificate file in PEM format. • KEY_FILE - Path to the private key file in PEM format. • CA_FILE - Path to the certificate authority (CA) file. The contents of the file might be the concatenation of an intermediate CA and the root CA (in that order). **Note:** When you set the `root-ca` certificate, omit the CA_FILE parameter.	Sets the specified certificate to be applied when **apicup subsys install** is run.
`apicup certs set SUBSYS CERT_NAME [KEY_FILE] [flags]`	KEY_FILE - The file that contains the encryption-secret for field level encryption in the management database. Applies only to the management subsystem . The certificate name is `encryption-secret`. The type is secure random bytes with a length of 128 bytes. For example, `apicup certs set mgmt1 encryption-secret /path/to/encryption-secret.bin`. **Note:** Do not specify any of the [CERT_FILE KEY_FILE CA_FILE] parameters when you set the encryption-secret.	Applies the `encryption-secret` when the management subsystem is installed.
`flags` • `--clear` • `--help`	Flags are: • --clear - Clears the specified certificate. For example, `apicup certs set mgmt1 encryption-secret --clear` • --help - Displays help for the command.	The specified certificate is cleared. When you make configuration changes such as changing endpoints, the corresponding certificate must be cleared so that a new certificate can be set.

Viewing details of a certificate

Run the `apicup certs get` command:

Table 2. apicup certs get

Command	Values	Result
`apicup certs get SUBSYS CERT_NAME [flags]`	Parameters are: • SUBSYS - name of the subsystem to which the certificate applies • CERT_NAME - name of the certificate to retrieve; see Certificate reference VMware for a list of certificates	Returns the specified certificate for the specified subsystem.
`flags` • `--output string` • `--type string` • `--help`	Flags are: • --output string- Specify a file for the retrieved values, or specify "-" to send to stdout. The default is "-" to send to stdout. For example, `apicup certs get mgmt1 --output myCertsFile` • --type string - Returns only the specified type. If not specified, the type is cert. For example, `apicup certs get mgmt1 --type ca` • --help - Displays help for the command.	• For --output: The specified certificate is retrieved and sent to stdout or saved to the specified file • For --type: Certificates are retrieved that match the type specified. .

List the certificates for a subsystem

Run the command **apicup certs list**:

Table 3. apicup certs list

Command	Values	Result
`apicup certs list SUBSYS [flags]`	Parameters are: • SUBSYS - name of the subsystem for which you want to list certificates	Returns a list of certificates that are configured for the subsystem.
`flags` • `--help`	Flags are: • --help - Displays help for the command.	Help text is displayed.

Generate certificates

The `apicup certs generate` command generates and sets default certificates. The `generate` command generates and sets a certificate only if it is not already set.

Note: The **apicup subsys install** operation also generates and sets any unset certificates, but use the `apicup certs generate` command so that you can set them and review them before you install.

Table 4. apicup certs generate

Command	Values	Result
`apicup certs generate SUBSYS [flags]`	Parameters are: • SUBSYS - name of the subsystem for which you want to generate certificates	Generates certificates that are currently not set for the subsystem. Generates self-signed certificates.
`flags` • `--help`	Flags are: • --help - Displays help for the command.	Help text is displayed.

Certificate validation reference

Certificates validation errors reference table.

The following table lists the certificate validation errors that the **apicup certs** command might report.

Validation	Messages	Error	Action to take
Verify that the certificate is set properly.	`certificate <cert> not set`	The certificate is not set.	
	`unable to load cert <cert>`	The certificate is set but cannot be read.	
Verify certificate key usage (Extended Key Usage).	`unable to verify cert <cert>: missing key usage <n>`	The certificate is missing the required key usage.	Check Certificate reference VMware to confirm the EKU required for the certificate.
Verify the certificate signing CA. If available, the CA file is loaded. Then the certificate is verified against the provided CA file, including enforcement of Extended Key Usage.	`unable to parse CA to verify cert <cert>`	The CA file could not be parsed and loaded.	
	`unable to verify cert <cert>`	The certificate failed verification against the provided CA file.	One possible reason for receiving this error is that the correct EKU is missing.
Verify certificate hosts.	`unable to verify cert <cert>: missing <host>`	The certificate is not valid for the required host.	See Certificate reference VMware for the required hosts.
Verify that a certificate that is being used as a CA is a CA certificate.	`unable to verify cert <cert>: certificate is not a CA`	The certificate is not a valid CA.	
Verify client certificate match. The `portal-client`, and `analytics-ingestion-client` certificates are verified against the CA of `portal-admin-ingress`, and `analytics-ingestion-ingress`.	`a CA certificate must be provided for this certificate`	The CA certificate is missing for one of the `portal-admin-ingress`, and `analytics-ingestion-ingress`.	The common certificates `portal-client`, and `analytics-ingestion-client` must be set before you set any custom certificates.
	`client cert cannot be verified against provided CA certificate`	The verification failed.	

Use JWT security instead of mTLS between subsystems

If your network requires TLS termination on load-balancers located between your API Connect subsystems, then disable mTLS and enable JWT to provide application layer security.

JWT security provides application layer security and can be used instead of mTLS when there are load-balancers located between subsystems that require TLS termination. For more information about JWT security, see Enable JWT instead of mTLS.

1. To enable JWT and disable mTLS, first identify the JWKS URL from the management subsystem:

   ```
   apicup subsys get <management subsystem>

   ...
   jwks-url     https://appliance1.apic.acme.com/api/cloud/oauth2/certs  JWKS URL for Portal and analytics subsystems to
   validate JWT -- this is unsettable and is generated based on the platform-api endpoint
   ...
   ```

2. Disable mTLS and enable JWT on portal and analytics by setting the `mtls-validate-client` and `jwks-url` values with **apicup**:

   ```
   apicup subsys set <portal> mtls-validate-client=false
   apicup subsys set <portal> jwks-url=appliance1.apic.acme.com/api/cloud/oauth2/certs

   apicup subsys set <analytics> mtls-validate-client=false
   apicup subsys set <analytics> jwks-url=appliance1.apic.acme.com/api/cloud/oauth2/certs
   ```

3. Apply the change to portal and analytics with `apicup subsys install`:

   ```
   apicup subsys install <subsystem_name>
   ```

4. To enable JWT and disable mTLS on the gateway appliance, see *Configuring the API Connect gateway service* in the DataPower documentation.

Note: Do not disable mTLS without enabling JWT.

Credentials and encryption secrets reference

API Connect user credentials and database encryption secrets that are managed with the **apicup certs** command.

The **apicup certs** command is used to manage various credentials and encryption secrets that are used by API Connect management and portal subsystems.
Note: For details of the TLS certificates that you can manage with the **apicup certs** command, see Certificate reference VMware.

Management subsystem credentials and secrets

The following table lists all the secrets and credentials that are returned from the **apicup certs list <management subsystem>** command:

Table 1. Management subsystem credentials and secrets

Secret or credential name	Description
`atm-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`consumer-toolkit-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`consumer-ui-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`database-backup-auth-ssh-key`	SSH key for authentication with a remote SFTP backup server.
`designer-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`encryption-secret`	A 128-bytes long encryption secret that is used to do field level encryption in the management subsystem database. If you want to modify this credential, open a support case.
`juhu-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`toolkit-credential`	Credential used in internal API Connect communication. Do not modify this credential.
`ui-credential`	Credential used in internal API Connect communication. Do not modify this credential.

Portal subsystem credentials and secrets

The following table lists all the secrets and credentials that are returned from the **apicup certs list <portal subsystem>** command:

Table 2. Portal subsystem credentials and secrets

Secret or credential name	Description
`encryption-secret`	A 128-bytes long encryption secret that is used to do field level encryption in the portal subsystem database.
`original-encryption-secret`	A copy of the encryption-secret that was first used to generate the Kubernetes secret name. Do not modify this secret.
`site-backup-auth-ssh-key`	Authentication key for your remote portal backup SFTP server.

Advanced configuration for management subsystem

Specify advanced configuration for the management subsystem.

Note: VMware deployments include subsystem settings that cannot be reconfigured without a complete deployment. For example, for the Management Service these settings include hosts, endpoints, interfaces (`public-iface`, `traffic-iface`), IP ranges of the Kubernetes pod and the service networks (`k8s-pod-network`, `k8s-service-network`). Before reconfiguring, review the installation instructions for your subsystem to determine which settings are optional during initial installation and thus able to be updated later.
See:

- **Setting rate limits for public APIs on the management service**
  Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.
- **Configuring maximum size of client requests to the Management subsystem on VMware**
  You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.
- **Enabling API governance on VMware**
  You can optionally configure API governance in API Connect on VMware by enabling the governance microservice.
- **Configuring monetization on VMware**
  You can optionally configure monetization in API Connect on a VMware virtual machine by enabling the billing microservice.
- **Configuring use of an external NTP server**
  You can optionally configure an external NTP server for use by API Connect when deploying on a VMware virtual machine.
- **Overriding resources for Postgres components on VMware V10.0.6.0**
  You can optionally override the CPU and Memory resources used by postgres database components.
- **Enabling UI feature flags on VMware**
  Use a template override to update the installation CR with settings that control UI features.

Setting rate limits for public APIs on the management service

Describes the procedure for setting a rate limit for public APIs on the management service. Rate limits provide protection from DDoS (distributed denial of service) attacks.

Before you begin

These instructions assume you have the kubectl command-line tool installed. For more information, see https://kubernetes.io.

About this task

Rate limits can be set for public APIs on the management service. Rate limits on APIs help provide protection from DDoS (distributed denial of service) attacks. Without a rate limit, API calls from public APIs are unlimited.

The rate limit configuration requires that the header contains the actual client IP address. Any load balancer or proxy (for example, HAProxy) that is installed in front of the management service must be configured to pass the actual client IP address.

This procedure must be performed on a running API Connect deployment.

Rate limits are calculated as requests per seconds per client.

Note: If the rate limit has been reached on the management subsystem, the client will get an HTTP error: **`429 Too Many Requests`**.

Procedure

- Set a rate limit:
  1. Add the following entries to the extra-values-file:

     ```
     spec:
       template:
       - name: juhu
         containers:
         - name: juhu
           env:
           - name: RATE_LIMIT_PER_CLIENT
             value: "10"
           - name: LIMIT_REQUEST_OPTION
             value: "burst=10 nodelay"
     ```

     In this example, the first option sets the rate to 10 requests per second (10r/s). The second option allows 10 requests boosted without delay within < 1 second. You can customize as needed.

     The **`rateLimitPerClient`** property sets **`rate`**, and **`limitRequestOption`** sets **`[burst=number] [nodelay | delay=number]`** in the following nginx configuration:

     ```
     limit_req_zone key zone=name:size rate=rate;
     limit_req zone=name [burst=number] [nodelay | delay=number];
     ```

     Note that **`zone`** has been pre-defined and can't be configured. For more details, see the nginx doc http://nginx.org/en/docs/http/ngx_http_limit_req_module.html.

  2. Save the extra-values file with a filename of your choice, but it must have a .yaml or .yml extension.
  3. Use **`apicup`** to update the VMware **`extra-values-file`** attribute to point to the new or updated management subsystem extra-values file:

     ```
     apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
     ```

     Where:
     - *<mgmt_subsystem_name>* is the name of the management subsystem that you are configuring.
     - *<extra_values_file_path_and_name>* is the fully qualified path and name of your extra-values file.

     For example:

     ```
     apicup subsys set ds10-management extra-values-file /workspace/v10012/my_extra_values.yaml
     ```

  4. Verify that the configuration settings are valid in the apiconect-up-v10.yaml by running the following command:

     ```
     apicup subsys get <mgmt_subsystem_name> --validate
     ```

     The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.
  5. Run **`apicup`** to update the settings in the deployed Management subsystem.

     ```
     apicup subsys install <management-subsystem>
     ```

  6. Enter the following command to connect to the management subsystem using SSH:

     ```
     ssh ip_address -l apicadm
     ```

     You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.

  7. Select Yes to continue connecting, and once connected:

     ```
     sudo -i
     ```

  8. Validate that the juhu pod has restarted by listing the pods:

     ```
     kubectl get pods  grep juhu
     ```

  9. Check the AGE column to ensure a new juhu pod has started.
  10. If the older juhu pod is still running, delete it with the following command:

      ```
      kubectl delete pods <old-juhu-pod>
      ```

- Disable a rate limit:
  1. Remove the values from the extra-values-file.
  2. Run **`apicup`** to update the settings in the deployed Management subsystem.

     ```
     apicup subsys install <management-subsystem>
     ```

3. Enter the following command to connect to the management subsystem using SSH:

   ```
   ssh ip_address -l apicadm
   ```

   You are logging in with the default ID of *apicadm*, which is the API Connect ID that has administrator privileges.

4. Select Yes to continue connecting, and once connected:

   ```
   sudo -i
   ```

5. Validate that the juhu pod has restarted by listing the pods:

   ```
   kubectl get pods  grep juhu
   ```

6. Check the AGE column to ensure a new juhu pod has started.
7. If the older juhu pod is still running, delete it with the following command:

   ```
   kubectl delete pods <old-juhu-pod>
   ```

Configuring maximum size of client requests to the Management subsystem on VMware

You can configure the maximum allowed size of the client request body for requests made to the Management subsystem.

About this task

The default maximum size is 8 megabytes. You can increase this value.

Increasing the setting can be useful if you get errors when using the CLI to import a large API. Example error:

```
HTTP/1.1 413 Request Entity Too Large
```

Note:

- These instructions apply only to VMware installations. For Kubernetes installations, see Configuring maximum size of client requests to the Management subsystem.
- These instructions use an extra-values file. This file can be used to define CR overrides. Note that extra-values files for a CR can contain more than one override. If you already use an extra-values file, you can add the new override to it.
- All of the commands must be run in the `apicup` management project directory.

Procedure

1. Create, or edit if one already exists, a management subsystem extra-values file in the `apicup` management project directory, and enter the following configuration details:

   ```
   spec:
     template:
     - name: juhu
       containers:
       - name: juhu
         env:
         - name: CLIENT_MAX_BODY_SIZE
           value: "12m"
   ```

2. Save the extra-values file with a filename of your choice, but it must have a .yaml or .yml extension.
3. Use `apicup` to update the VMware `extra-values-file` attribute to point to the new or updated management subsystem extra-values file:

   ```
   apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
   ```

   Where:
   - *<mgmt_subsystem_name>* is the name of the management subsystem that you are configuring.
   - *<extra_values_file_path_and_name>* is the fully qualified path and name of your extra-values file.

   For example:

   ```
   apicup subsys set ds10-management extra-values-file /workspace/v10011/my_extra_values.yaml
   ```

4. Verify that the configuration settings are valid in the apiconect-up.yaml by running the following command:

   ```
   apicup subsys get <mgmt_subsystem_name> --validate
   ```

   The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.
5. Update the management VMware with the updated setting.

   ```
   apicup subsys install <mgmt_subsystem_name>
   ```

6. Monitor the health-check output until the management subsystem is healthy by running the following command:

   ```
   apicup subsys health-check <mgmt_subsystem_name>
   ```

   If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of `1`. The following output is an example of unhealthy output while the install is running:

```
Error: Cluster not in good health:
 ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
 ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
```

When all of the health criteria are successfully met, the command displays no output, and exits with a status of `0`.

Enabling API governance on VMware

You can optionally configure API governance in API Connect on VMware by enabling the governance microservice.

About this task

API governance is an optional add-on to IBM® API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process.

Note:

- These instructions apply only to VMware installations. For Kubernetes, OpenShift, and IBM Cloud Pak for Integration installations, see Enabling API governance on Kubernetes.
- API governance rulesets cannot be added to your deployment until the governance microservice is enabled.
- All of the commands must be run in the `apicup` management project directory.

To enable or disable the governance microservice, see the following instructions:

- **Enabling the governance microservice**
- **Disabling the governance microservice**

After the governance microservice is enabled, API governance resources can be created. For more information, see Configuring API governance in the Cloud Manager, and Configuring API governance in the API Manager.

Procedure

- **Enabling the governance microservice**
  1. Open your API Connect installation project directory.
  2. Run the following `apicup` command to enable the governance microservice:

     ```
     apicup subsys set <mgmt_subsystem_name> governance-enabled=true
     ```

     Where *`<mgmt_subsystem_name>`* is the name of the management subsystem that you are configuring.

  3. Install the new setting for the governance microservice by running the following command:

     ```
     apicup subsys install <mgmt_subsystem_name> --debug
     ```

     Including the `--debug` option enables the debug output for the command.

  4. Monitor the health-check output until the management subsystem is healthy by running the following command:

     ```
     apicup subsys health-check <mgmt_subsystem_name>
     ```

     If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of `1`. The following output is an example of unhealthy output while the install is running:

     ```
     Error: Cluster not in good health:
      ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
      ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
     ```

     When all of the health criteria are successfully met, the command displays no output, and exits with a status of `0`.
- **Disabling the governance microservice**
  1. Open your API Connect installation project directory.
  2. Run the following `apicup` command to disable the governance microservice:

     ```
     apicup subsys set <mgmt_subsystem_name> governance-enabled=false
     ```

     Where *`<mgmt_subsystem_name>`* is the name of the management subsystem that you are configuring.

  3. Install the new setting for the governance microservice by running the following command:

     ```
     apicup subsys install <mgmt_subsystem_name> --debug
     ```

     Including the `--debug` option enables the debug output for the command.

Results

Note that when the governance microservice is enabled, there are a number of new deployments, jobs, and pods in the `ManagementCluster` namespace. These Kubernetes governance resources have names containing either `compliance-service` or `compliance-ui`. For example:

```
kubectl get pods -n apic | grep compliance
management-compliance-service-f6cdf95fc-t4qkx                     1/1     Running     0          127m
management-compliance-ui-59897fcc4-zm25v                          1/1     Running     0          126m
management-up-compliance-service-data-populate-0-to-1-t2f4d       0/1     Completed   1          132m
management-up-compliance-service-schema-0-to-1-2lkqq              0/1     Completed   0
```

Configuring monetization on VMware

You can optionally configure monetization in API Connect on a VMware virtual machine by enabling the billing microservice.

Before you begin

Important: If the optional billing microservice is enabled, you must increase the memory for the API Manager for VMware component by 0.5 GB.

About this task

API Connect includes a subscription billing microservice that allows API providers to define pricing plans in their API Products, and monetize their API offerings. If a Product contains a pricing plan, API Consumers must enter their payment information into the Developer Portal before they can subscribe to that plan. API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Note:

- These instructions apply only to VMware installations. For Kubernetes installations, see Configuring monetization on Kubernetes.
- To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See Firewall requirements on Kubernetes and Firewall requirements on VMware for more information about this requirement.
- Provider organizations cannot add any billing integrations until the billing microservice is enabled.
- After provider organizations have added billing integrations, the billing microservice must not be disabled.
- All of the commands must be run in the `apicup` management project directory.

To enable or disable the billing microservice, you must configure a management subsystem extra-values file. See the following instructions:

- **Enabling the billing microservice**
- **Disabling the billing microservice**

Procedure

- **Enabling the billing microservice**
  1. Create, or edit if one already exists, a management subsystem extra-values file in the `apicup` management project directory, and enter the following configuration details:

     ```
     spec:
       billing:
         enabled: true
     ```

  2. Save the extra-values file with a filename of your choice, but it must have a .yaml or .yml extension.
  3. Update the VMware `extra-values-file` attribute to point to the new or updated management subsystem extra-values file.
     Run the following `apicup` command to insert the location of the extra-values file:

     ```
     apicup subsys set <mgmt_subsystem_name> extra-values-file <extra_values_file_path_and_name>
     ```

     Where:
     - *<mgmt_subsystem_name>* is the name of the management subsystem that you are configuring.
     - *<extra_values_file_path_and_name>* is the fully qualified path and name of your extra-values file.

     For example:

     ```
     apicup subsys set ds10-management extra-values-file /workspace/v10011/monetization.yaml
     ```

  4. Verify that the configuration settings are valid in the apiconect-up.yaml by running the following command:

     ```
     apicup subsys get <mgmt_subsystem_name> --validate
     ```

     The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.
  5. Update the management VMware with the updated billing value.
     Install the new setting for the billing microservice by running the following command:

     ```
     apicup subsys install <mgmt_subsystem_name> --debug
     ```

     Including the `--debug` option enables the debug output for the command.
  6. Monitor the health-check output until the management subsystem is healthy by running the following command:

     ```
     apicup subsys health-check <mgmt_subsystem_name>
     ```

     If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of `1`. The following output is an example of unhealthy output while the install is running:

     ```
     Error: Cluster not in good health:
      ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
      ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
     ```

     When all of the health criteria are successfully met, the command displays no output, and exits with a status of `0`.
     The billing microservice is now enabled, and provider organization owners can create billing integration resources for API providers in the API Manager UI.
     For more information, see Monetizing your Products.

- **Disabling the billing microservice**

Important: After provider organizations have added billing integrations, the billing microservice must not be disabled.

1. Edit the management subsystem extra-values file in the `apicup` management project directory, and change the `enabled` configuration option to `false`:

   ```
   spec:
     billing:
       enabled: false
   ```

2. Save the changed extra-values file.
3. Verify that the configuration settings are valid in the apiconect-up.yaml by running the following command:

   ```
   apicup subsys get <mgmt_subsystem_name> --validate
   ```

   The output lists each setting and adds a check mark after the value once the value is validated. If the setting lacks a check mark and indicates an invalid value, reconfigure the setting.
4. Update the management VMware with the altered billing value.
   Install the updated setting for the billing microservice by running the following command:

   ```
   apicup subsys install <mgmt_subsystem_name> --debug
   ```

   Including the `--debug` option enables the debug output for the command.
5. Monitor the health-check output until the management subsystem is healthy by running the following command:

   ```
   apicup subsys health-check <mgmt_subsystem_name>
   ```

   If one or more of the health criteria are not met, the command stops processing and displays a message with the failure, and exits with a status of `1`. The following output is an example of unhealthy output while the install is running:

   ```
   Error: Cluster not in good health:
    ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
    ManagementCluster (current ha mode: active) is not ready | State: 15/17 Phase: Pending
   ```

   When all of the health criteria are successfully met, the command displays no output, and exits with a status of `0`.
   The billing microservice is now disabled.

Related information

- Monetizing your Products
- stripe.com

Configuring use of an external NTP server

You can optionally configure an external NTP server for use by API Connect when deploying on a VMware virtual machine.

About this task

Secure communication between API Connect subsystems relies on the system time being in sync on all hosts. For example, time stamps are checked to ensure that certificates are valid. When API Connect is deployed behind a firewall that blocks access to the internet, the API Connect subsystems cannot by default access a Network Time Protocol (NTP) server.

You can use an additional cloud-init file to manually specify an NTP server for use by the subsystems. Complete the following steps.

Procedure

1. Create the cloud-init file extra values file, and enter the configuration details that you want to overwrite. For example:

   ```
   ntp:
     enabled: true
     ntp_client: systemd-timesyncd
     servers:
       - time.google.com
   ```

2. Use `apicup` to specify the cloud-init file.
   Syntax:

   ```
   apicup subsys set <subsys> additional-cloud-init-file <path-to-cloud-init-file>
   ```

   Example:

   ```
   apicup subsys set mgmt additional-cloud-init-file myCloudInitFile.yaml
   ```

3. Install the subsystem. Note that the output directory must be empty:

   ```
   apicup subsys install mgmt --out mgmtplan-out
   ```

4. Deploy the VMware image (.ova) with the ISO file that is generated.
   To review the deployment steps, see Deploying the Management virtual appliance.

5. Verify that the correct NTP server is being used:

   ```
   journalctl -u systemd-timesyncd
   ```

   Example output for the NTP server that was set in Step 1:

```
Nov 05 21:09:24 h-apicdev-4 systemd[1]: Starting Network Time Synchronization...
Nov 05 21:09:24 h-apicdev-4 systemd[1]: Started Network Time Synchronization.
Nov 05 21:09:24 h-apicdev-4 systemd-timesyncd[1697]: Synchronized to time server 216.239.35.8:123 (time.google.com).
```

Overriding resources for Postgres components on VMware V10.0.6.0

You can optionally override the CPU and Memory resources used by postgres database components.

About this task

Note: The information in this topic applies to V10.0.6.0 only.
There are three postgres components:

- `postgres`
- `pgBouncer`
- `pgBackRest`

The default resources used in `postgrespgBouncer` and `pgBackRest` components are:

Table 1. Postgres component default resources

Resource	Postgres DB	pgBouncer	pgBackRest
Memory Request	512Mi	24Mi	48Mi
Memory Limit	8Gi	128Mi	128Mi
CPU Request	600m	80m	80m
CPU Limit	1500m	200m	200m

Procedure

To override resources, complete the following procedure

1. Create an extra-values yaml file, or add values to an existing extra-values file.
   - Ensure that you specify the name of the container for the resource you want to override. Names:
     - `postgres`
     - `pgBouncer`
     - `pgBackRest`
   - You can override multiple postgres components in parallel. Adding one container section under `mgmt-pgcluster` template.
   - You can override just a single value, such as CPU limit.

   Examples:
   - Example yaml, to replace all `postgres` resources:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: postgres
           resources:
             requests:
               cpu: 1500m
               memory: 8Gi
             limits:
               cpu: 600m
               memory: 512Mi
     ```

   - Example yaml, to replace all `pgBouncer` resources:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: pgBouncer
           resources:
             requests:
               cpu: 200m
               memory: 128Mi
             limits:
               cpu: 300m
               memory: 256Mi
     ```

   - Example yaml,to replace CPU limit of `postgres` database:

     ```
     spec:
       template:
       - name: mgmt-pgcluster
         containers:
         - name: postgres
           resources:
             limits:
               cpu: 1000m
     ```

   - Example yaml, to replace CPU limits of `postgres` database and `pgBouncer`:

     ```
     spec:
       template:
     ```

```
  - name: mgmt-pgcluster
    containers:
    - name: postgres
      resources:
        limits:
          cpu: 1000m
    - name: pgBouncer
      resources:
        limits:
          cpu: 1000m
```

2. Set the extra values file:

```
apicup subsys set <SUBSYSTEM_NAME> extra-values-file=<PATH_TO_EXTRA_VALUES_FILE>
```

3. Apply the overrides:

```
apicup subsys install <SUBSYSTEM_NAME>
```

4. Check health of cluster:

```
apicup subsys health-check <SUBSYSTEM_NAME>
```

Results

The postgres components will restart with updated values.

Enabling UI feature flags on VMware

Use a template override to update the installation CR with settings that control UI features.

About this task

You can add a template override to the Management subsystem CR and use it to control whether specific features are available in the UI. Some features can be optionally disabled by users (by disabling them in the browser) provided you have enabled the feature for the cluster as a whole. Table 1 describes the available feature flags.

Table 1. Flags and settings for controlling UI features

Feature ID	Default Value	Partial Override For	Description
`ffAdvancedPublish`	`true`		Enable advanced publish option (deprecated)
`allOmniSearch`	`true`		Enables OmniSearch for all lists. This flag is used in conjunction with other flags.
`draftsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for drafts list
`catalogProductsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for catalog
`orgsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for org lists
`appsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for apps
`subscriptionsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for subscriptions, if disabled, subscription lists will be removed from catalog list
`requestedApprovalsOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for requested approvals
`approvalTasksOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for approvals
`productApisOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for apis within products
`membersOmniSearch`	`true`	`allOmniSearch`	Enable OmniSearch for member lists

Procedure

1. Add the template override to the `management-extra-values.yaml` file.
2. In the file, append the feature-flag definition to the end of the `spec:` section, making sure to adhere to the spacing used in the file.
   The following example enables the OmniSearch feature on all possible UI pages.

```
spec:
  template:
    - name: ui
      containers:
      - name: ui
        env:
          - name: APIC_UI_FEATURES
            value: allOmniSearch=always
```

   `APIC_UI_FEATURES` is a URL-encoded string. Each parameter is formed as `featureId=$featureValue`, and you can list multiple parameters with the `&` delimiter (`featureId=$featureValue&featureId=$featureValue`).

   Use the `featureId` values from Table 1. For each feature, specify one of the following `featureValue` settings:

   - `true` - On by default, but can be overridden by the user, for their browser alone
   - `false` - Off by default, but can be overridden by the user, for their browser alone
   - `never` - Off regardless of browser setting, disabled in feature flag page
   - `always` - On regardless of browser setting, disabled in feature flag page
3. Save the file.
4. Run the following command to update the deployment:

```
apicup subsys set <management_subsystem> extra-values-file <path_to_management-extra-values.yaml_file>
```

The cluster is updated to enable or disable features as specified.

Configuring rate limits on the gateway nodes on VMware

Create an env.yaml file on each Gateway node to configure the default rate limit.

The Gateway uses scale limits (instead of strict limits) by default to enforce rate limits. This improves the gateway's performance by providing the ability to handle large amounts of rate-limited traffic.

1. To configure the rate limit, create a file called env.yaml with the settings provided in Table 1:

Table 1. Rate limit settings

Setting	Type	Default	Optoins	Comments
`USE_RATE_LIMIT_DEFINITION`	boolean	`true`	• `true` Use scale limits • `false` Use strict limits	Determines the type of rate limiting to use on the Gateway node.
`RATE_LIMIT_UPDATE_MODE`	enumerated	`prior`	• `none`No limits in a plan are incremented after a limit is reached. • `all` All limits in a plan are incremented after a limit is reached. • `prior` Retains the behavior from pre-V10 releases: When the plan rate limit is reached, all limits in the plan that are sorted before that plan limit will be incremented.	Determines how the counts within a plan are incremented when a rate limit is hit. Applies only to scale limits.
`RATE_LIMIT_SORT_BY_DURATION`	boolean	`true`	• `true` Sort the rate limits by duration. • `false` Revert to the previous sort order.	Determines the sort order for rate limits in a plan, which in turn determines the sequence in which they are applied.
`RATE_LIMIT_USE_OFFSET`	boolean	`false`	• `true` Prevent all rate limit counts for an interval from being reset at the same time. • `false` Allow all rate limit counts for an interval to be reset at the same time.	Allow or prevent all of the rate limit counts for an interval from being reset at the same time. Preventing the simultaneous reset protects backends from being saturated with requests by resetting the counts for different subscribers of a plan at different points with in the interval. Applies only to scale limits.
`RATE_LIMIT_DEFINITION_RESPONSE_HEADERS`	boolean	`true`	• `true` Show the `x-rate-limit` headers. • `false` Hide the `x-rate-limit` headers.	Show or hide the `x-rate-limit` headers in an API response. Applies only to scale limits.

2. Save the file in the local:/ directory on the Gateway node so that the file persists on reboots of both physical and virtual machines. The settings apply only to the node where the file is stored, so each node in the Gateway cluster requires its own copy of the file.

Backing up and restoring on VMware

Backup and restore API Connect on VMware.

Backups are for recovery of the API Connect subsystems in the same environment that the backups were taken. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see Migrating from v10 to v10 on a different form factor.

Note: For information about how to maintain a two data center deployment, including backup and restore considerations, see Maintaining a two data center deployment.

- **Backing up and restoring the Management subsystem**
  Backup your management project directory and configure scheduled database backups.

- **Backing up and restoring the Management subsystem V10.0.6.0**
  You can back up and restore your Management subsystem in your VMware environment.
- **Backing up and restoring the Developer Portal**
  You can backup and restore a Developer Portal in a VMware environment.
- **Backing up and restoring the Analytics database on VMware**
  The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

Backing up and restoring the Management subsystem

Backup your management project directory and configure scheduled database backups.

Management subsystem backups are intended for restoration in the same environment where they were taken, or a different environment that has the same network configuration, endpoints, and deployment profile.

A management subsystem backup consists of:

Project directory backup
: The original project directory that was created with the **apicup** command during the initial product installation is essential for disaster recovery, see: First steps for deploying in a VMware environment. The project directory contains the YAML files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the management database on a deployment that uses a new or different project directory.

  Take a new backup of your project directory every time you make any configuration changes with the **apicup** command. Store your project directory backups at a remote location.

  Ensure that you keep all files related to your management subsystem in the project directory, so they are included in your project directory backup. For example:

  - Any TLS certificate **.pem** files that were used to customize management subsystem certificates.
  - Your management extra-values.yaml file, if you created one, for example: Configuring maximum size of client requests to the Management subsystem on VMware.

Management database backup
: Your management database backup contains everything that you create and configure in the Cloud Manager and API Manager UI (APIs, products, TLS profiles, user registries, and so on), or with the toolkit CLI or REST API. You configure your management subsystem to run scheduled backups. You can also take backups on-demand. The configuration of your database backups, such as S3 object store credentials, are defined by the management configuration YAML file in your project directory. Therefore, to restore the management database backup after a disaster, you must first restore your project directory.

Configuring management subsystem backups

1. Take a backup of your project directory manually, and store the backup at a remote location. Whenever you update your management subsystem with the **apicup** command, you should take a new backup of this directory.
2. Configure management database backups: Configuring management database backups.

Snapshots

Take a Virtual Machine (VM) snapshot of all your management VMs; see Using VM snapshots for infrastructure backup and disaster recovery. Taking snapshots does require a brief outage while all the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can provide faster recovery when compared to redeploying OVAs and restoring from a database backup.

Important: VM snapshots are not an alternative to the standard API Connect backups that are described in this section. Do not rely solely on VM snapshots for your backups.

- **Configuring management database backups**
  Use **apicup** to configure the database backup settings, before or after management subsystem installation.
- **Running an on-demand backup of the management database**
  Create a management database backup at any time, with **apicup**.
- **Restoring the management database backup**
  Use **apicup** to list backups and restore a selected backup of the management subsystem
- **Recovering from a disaster**
  Recover the management subsystem in a VMware environment.

Configuring management database backups

Use **apicup** to configure the database backup settings, before or after management subsystem installation.

About this task

- You use the **apicup** utility to configure backup settings for the management subsystem.
- Backups can be scheduled or generated manually (on-demand).
- You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.
- You must have a remote S3 object-store available to store your management database backups.
  Types of S3 backups supported:
  - IBM Cloud Object Storage
  - AWS S3
  - Any custom S3, such as minIO

- The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: **`x509: certificate signed by unknown authority`**
- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.
- You can modify any of the backup settings at any time, but if you change the **`database-backup-auth-user`** or **`database-backup-auth-pass`** settings, you must also run:

  `apicup subsys db-reload <subsystem_name>`

  after applying the new settings with:

  `apicup subsys install <subsystem_name>`

Procedure

1. Use **`apicup`** to specify backup settings:

   `apicup subsys set <subsystem_name> <setting>=<value>`

Table 1. Configuration settings for backup of management subsystem database

Parameter	Description
`database-backup-auth-pass`	The password for the server specified in **`database-backup-host`**. This is the S3 Secret Access Key parameter. The password is converted and stored in Base64 encoded format. For example: **`apicup subsys set <subsystem_name> database-backup-auth-pass '<password>'`** Note that you cannot use the `=` sign to assign the password to **`database-backup-auth-pass`**. If you change this value later, to apply the change you must run **`apicup subsys db-reload <subsystem_name>`** after **`apicup subsys install <subsystem_name>`**
`database-backup-auth-user`	The username for the server specified in **`database-backup-host`**. This is the S3 Secret Key ID. **`apicup subsys set <subsystem_name> database-backup-auth-user <username>`** If you change this value later, to apply the change you must run **`apicup subsys db-reload <subsystem_name>`** after **`apicup subsys install <subsystem_name>`**
`database-backup-host`	The backups host. Specify S3 endpoint with the corresponding S3 region in the format **`<S3endpoint>/<S3region>`**: • **`apicup subsys set <subsystem_name> database-backup-host=s3.ibm.com/eu-west-2`** • **`apicup subsys set <subsystem_name> database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard`**
`database-backup-path`	The path to the location of the backup. Specify the name of your S3 bucket to store backup data, in the format **`bucket-name/folder`**. The use of subdirectories in the bucket name is not supported. When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same S3 bucket name in the database backup configurations. Each management subsystem must use a unique S3 bucket name. Ensure that **`bucket-name/folder`** is empty. **`apicup subsys set <subsystem_name> database-backup-path=bucket-name/apic-backup`**
`database-backup-schedule`	Cron schedule for performing automatic backups. The format for the schedule is: • `******` • `------` • |||||| • ||||| +----- day of week (0 - 6) (Sunday=0) • |||| +------- month (1 - 12) • ||| +--------- day of month (1 - 31) • || +----------- hour (0 - 23) • | +------------- min (0 - 59) • +--------------- sec (0 - 59) The default is an empty string, which means no backups are taken. The timezone for backups is UTC. This example will schedule a back up daily at 1am. **`apicup subsys set <subsystem_name> database-backup-schedule="0 0 1 * * *"`**
`backup-certs`	**Optional**. Backup certificates to use for TLS communication between API Connect and the backup server. For example: **`apicup subsys set <subsystem_name> backup-certs=<certificate_name>`**

Parameter	Description
`database-backup-s3-uri-style`	**Optional**: Specifies whether the URI is in the form of `host` or `path`. `apicup subsys set <subsystem_name> database-backup-s3-uri-style='<host|path>'` Supported values: • `host` • `path`
`database-backup-repo-retention-full`	The number of days of backups to retain. For example, if `database-backup-repo-retention-full` is set to 30, then backups for the last 30 days are retained. Older backups are deleted. After a restore, the count of the number of days is reset to zero. Minimum value: 1 Maximum value: 9999999 Default: none

2. Confirm that the backup settings pass validation:

   ```
   apicup subsys get <subsystem_name> --validate
   ```

   If you are configuring the backup settings before install, then skip the remaining steps and continue with the installation process. Verify that your backups are working after installation.
3. Apply the settings:

   ```
   apicup subsys install <subsystem_name>
   ```

4. Ensure that the changes were applied successfully:

   ```
   apicup subsys health-check <subsystem_name>
   ```

5. Take a new backup of your project directory.
   Your backup configuration settings are stored in your project directory. Take a new backup of your project directory every time you update your backup settings or make any other configuration updates with the **apicup** command.

6. Verify that your backup settings are correct by running an on-demand backup: Running an on-demand backup of the management database.

Running an on-demand backup of the management database

Create a management database backup at any time, with **apicup**.

Before you begin

- Database backup settings for the management subsystem must be configured. See Configuring management database backups.
- Take a backup of the portal database at the same time as you take a backup of the management database. The portal database and management database must remain consistent with each other.
- Do not attempt to run more than one management database backup at a time. Check that no backups are running:

  ```
  apicup subsys list-backups <management subsystem name>
  ```

  Check that the scheduled backup is not configured to run before you expect your on-demand backup to finish:

  ```
  apicup subsys get <management subsystem name> database-backup-schedule
  ```

Procedure

1. Check that your management subsystem is healthy:

   ```
   apicup subsys health-check <management subsystem name>
   ```

   When successful, the command returns silently with an exit code of `0`.

2. Use the **apicup** command to start the backup process:

   ```
   apicup subsys backup <management subsystem name>
   ```

   To view available flags, use `-h`:

   ```
   apicup subsys backup management -h
   Backup management subsystem
   Usage:
     backup SUBSYS [flags]Flags:
         --debug                  Enable debug logging
     -h, --help                   help for backup
         --wait                   Wait for the operation to complete or fail.
         --wait-timeout duration  Command timeout in seconds. (default 40s)
   ```

3. Monitor the backup process with:

   ```
   apicup subsys list-backups <management subsystem name>
   ```

   Example output:

   ```
   NAME                AGE   CLUSTER                         PHASE       ERROR
   mgmt-backup-qdbfn   65m   boaz-auto1-87f65bf8-site1-db    completed
   ```

For more information about the backup, use the `detailed-output` flag:

```
apicup subsys list-backups <management subsystem name> --detailed-output=true
```

Restoring the management database backup

Use `apicup` to list backups and restore a selected backup of the management subsystem

Before you begin

- When you restore the management subsystem database, you must then restore the portal subsystem database from a backup that was taken at the same time. The database backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.
- Restoring the management database requires downtime and is a destructive process that deletes current data and copies backup data. During the restoration process, external traffic must be stopped.
- To restore the management database, you must use the original project directory that was created with `apicup` during the initial product installation. You cannot restore the database without the initial project directory. If necessary, restore the project directory from your Project directory backup.

About this task

Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For more information about the registration credentials, see Changing the registration client_id and client_secret for applications.

Procedure

1. List the available backups:

   ```
   apicup subsys list-backups <management subsystem name>
   ```

   Example output:

   ```
   # apicup subsys list-backups mgmt
   NAME                AGE   CLUSTER                          PHASE       ERROR
   mgmt-backup-qdbfn   65m   boaz-auto1-87f65bf8-site1-db   completed
   mgmt-backup-qdfdd   24h   boaz-auto1-87f65bf8-site1-db   completed
   ```

   Take note of the name of the backup that you want to restore.
   If the backup you want to restore is not listed, but you know where it is on the S3 object-store, then you can restore by using the backup ID instead of the backup name, see step 3.

2. Restore the backup, by using the backup name identified in step 1.

   ```
   apicup subsys restore <management subsystem name> --name <backupname> <flags>
   ```

   Available flags:

   ```
   apicup subsys restore mgmt --help
   Restore management subsystem

   Usage:
     restore SUBSYS [flags]

   Flags:
         --debug                                Enable debug logging
     -h, --help                                 help for restore
         --host <host>/<myregion>               FQDN for DB backups server in format <host>/<myregion>
         --id string                            ID of the backup to restore
         --name string                          Name of the backup to restore
         --password string                      Password for DB backups server
         --path string                          Path for DB backups server
         --username string                      Username for DB backups server
         --wait                                 Wait for the operation to complete or fail.
         --wait-timeout duration                Command timeout in seconds. (default 40s)
   ```

   Example:

   ```
   apicup subsys restore boaz-ova2-management --name mgmt-backup-qdbfn

   Starting restore...
   please check later for completion status of restore
   ManagementRestore is RestoreInProgress
   ```

3. Restore the backup, by using a backup ID.
   If you do not have a backup name for the backup you want to restore, you can use a backup ID. Look on your S3 object-store to identify the ID of the management database backup that you want to restore.
   In your S3 store, management database backups appear as folders that use a datestamp for their name, and are located inside the backup path. Example of management subsystem and its full path on an S3 store:

   ```
   apic-mgmt-backups/impr-restore/m1-65288d1d-db-2023-09-14T01:30:40Z/base/20230914T013040
   ```

   - `apic-mgmt-backups/impr-restore/m1-65288d1d-db-2023-09-14T01:30:40Z` is the backup path. The datestamp `2023-09-14T01:30:40Z` indicates when the management database was created (not when the backup was created). The management database is created on initial installation, and recreated when a backup restore is run. The path with the most recent datestamp should contain the most recent backups.
   - `20230914T013040` is the backup ID. The format indicates the date and time that the backup was taken.

- `base` is an autogenerated path segment and is ignored.

When you restore with a backup ID, you must include the S3 host, path, username, and password as flags in the restore command:

```
apicup subsys restore <management subsystem name> --id <backup ID> --host <S3 hostname> --username <S3 username> --
password <S3 password> --path <S3 backup path>
```

For example:

```
apicup subsys restore boaz-ova2-management --id 20230725T162548 --host s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-gb
--username 'a05e7aace2ab43b5941326429c8c071c' --password '3e510a1d9b06189b57334d064badf6abc39ab5a51d9bdcf7' --path
upgrade-test/boaz-ova1-management-22/management-site1-db-2023-11-16T12:30:02Z

Starting restore...
INFO[0000] Creating generic secret mgmt-restore-auth-secret.
please check later for completion status of restore
ManagementRestore is RestoreInProgress
root@apicdev3067:/home/apicadm/boaz-ova2# k get mgmtr
NAME                  STATUS              MESSAGE
BACKUP               CLUSTER                PITR                  AGE
NAME                  STATUS     MESSAGE                                                  BACKUP              CLUSTER
PITR                   AGE
mgmt-restore-t6ss5    Complete   Restore process completed (DB Restore + DRR job)    mgmt-backup-cdkv6   boaz-ova2-
management   2023-07-25T16:25:58   127m
mgmt-restore-tgxl2    Complete   Restore process completed (DB Restore + DRR job)                        boaz-ova2-
management
```

4. Monitor the restore:

   ```
   apicup subsys list-restores <management subsystem name>
   ```

5. When the management database restore shows the status `RestoreSuccessful`, restore your portal database backup: Restoring the Developer Portal subsystem.

Recovering from a disaster

Recover the management subsystem in a VMware environment.

About this task

Recovering the management subsystem involves re-creating your management ISO files, redeploying your management VMs with the new ISO files, and then restoring the management database from your backup.

Important points:

- In a clustered deployment, if any one VM is corrupted then all the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.
- When you restore the management subsystem database, you must then restore the portal subsystem database from a backup that was taken at the same time. The database backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.
- The disaster recovery process is intended for restoration of API Connect in the same network environment with the same endpoints. If you want to change API Connect endpoints or migrate to a different form-factor, see Migrating from v10 to v10 on a different form factor.

Procedure

1. Use the same project directory that you used for your original deployment, or a restore of your Project directory backup.
2. Create your ISO files:

   ```
   apicup subsys install <subsystem name> --out <output directory>
   ```

   where *<output directory>* is the directory where the ISO files are to be created.
   Note: If your original ISO files are still available and you haven't upgraded from the original installation, you can reuse them. However, if you have upgraded your original deployment, you must create new ISO files using the version of **apicup** that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.7.x OVAs with ISO files that were created with **apicup** v10.0.6.0.
3. Deploy your new management VMs: Deploying the Management subsystem OVA file.
4. Verify your new management deployment, but do not log in to the Cloud Manager UI or attempt to configure or change any other settings: Verify installation of the Management subsystem.
5. Restore your management database backup, using backup ID. See step 3 on Restoring the management database backup.

Backing up and restoring the Management subsystem V10.0.6.0

You can back up and restore your Management subsystem in your VMware environment.

About this task

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem

- You use the `apicup` utility to configure backup settings for the management subsystem.
- Database backups can be used to restore the database for disaster recovery.
- Backups can be scheduled or generated manually (on-demand). You can then list what backups are available and then use one to restore your Management subsystem if required.
- Postgres replica pods depend on a successful completed backup. If backup configuration is not correct, replica pods won't come up.

- Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.

Note: You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.

Types of backups supported:

- Cloud-based S3 storage
  - IBM Cloud Object Storage
  - AWS S3
  - **v10.0.2.0 or later:** Any custom S3, such as minIO
- SFTP
- Local Backups

- **Configuring backup settings during initial installation of the management subsystem V10.0.6.0**
  You can use the `apicup` utility to configure backup settings during initial installation of the management subsystem
- **Verify configuration for s3 backup V10.0.6.0**
  Verify that the configuration for s3 backups succeeded.
- **Reconfiguring backup settings for the management subsystem V10.0.6.0**
  You can use the `apicup` utility to add or reconfigure backup settings for the management subsystem, after completion of initial installation.
- **Backing up the management subsystem V10.0.6.0**
  Use apicup to create a manual backup of the management subsystem
- **Restoring the management subsystem V10.0.6.0**
  Use `apicup` to list backups and restore a selected backup of the management subsystem
- **Troubleshooting management subsystem backups on VMware V10.0.6.0**
  You can troubleshoot failures of the management subsystem backups.

Configuring backup settings during initial installation of the management subsystem V10.0.6.0

You can use the `apicup` utility to configure backup settings during initial installation of the management subsystem

Before you begin

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
Review Backing up and restoring the Management subsystem V10.0.6.0.
Note: You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.

About this task

- During an initial installation, use these steps to configure the backup settings prior to creating the initial management ISO.
- You can configure scheduled backups, or generate manual (on-demand) backups.
- Changing backup settings results in database downtime, unless `database-backup-protocol=sftp`.
- Backup to S3 providers of type `custom` is supported in v10.0.2.0 or greater. For `custom` S3, provide values for `database-backup-certs` and `database-backup-s3-uri-style`.
- OpenSSH key authentication for SFTP is supported.
- When you specify a filepath in a backup setting, be sure to escape any blank spaces in the path.

Important: For S3 backups of the management subsystem, do not use retention features provided by Cloud-based S3 storage providers. Use of such features can result in periodic deletion of archived backups, which can cause backup corruptions that can cause database restores to fail.

Procedure

1. Use `apicup` to specify backup settings:
   - For most settings, use `apicup subsys`:

     ```
     apicup subsys set <subsystem_name> <setting>=<value>
     ```

   - When configuring ssh-key authentication for SFTP, use `apicup certs`

     ```
     apicup certs set <subsystem_name> database-backup-auth-ssh-key <ssh-key-file-path>
     ```

Table 1. Configuration settings for backup of management subsystem

Parameter	Description

Parameter	Description
database-backup-protocol	The type of the backup. Supported types: • **objstore** **apicup subsys set mgmt database-backup-protocol=objstore** • **sftp** **apicup subsys set mgmt database-backup-protocol=sftp** The default value is **sftp**. • **local** - Local storage backups **apicup subsys set mgmt database-backup-protocol=local** Note: • The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: **x509: certificate signed by unknown authority**.
database-backup-auth-pass	The password for the server specified in **database-backup-host**. If using object store, this would be the S3 Secret Access Key parameter. The password will be stored in Base64 encoded format. For example: **apicup subsys set mgmt database-backup-auth-pass '<*password*>'** Note that you cannot use the `=` sign to assign the password to **database-backup-auth-pass**.
database-backup-auth-user	The username for the server specified in **database-backup-host**. If using object store, this would be the S3 Secret Key ID. **apicup subsys set mgmt database-backup-auth-user <*user_name*>**
database-backup-auth-ssh-key	Optional: OpenSSH private key for DB backups using SFTP only. Changing this value will restart the DB. Important: • Use **apicup certs** to set this value. Do not use **apicup subsys**. • Only OpenSSH keys are supported.[1] **apicup certs set mgmt database-backup-auth-ssh-key <*ssh private key filepath*>**
database-backup-s3provider	When using **objstore** type backup, specify the name of the S3 provider to use. Specify one of the supported values: • **ibm** • **aws** • **custom** - **v10.0.2.0 or later** Not required for local backups or SFTP backups. **apicup subsys set mgmt database-backup-s3provider=ibm**
database-backup-host	The backups host: • For **objstore** type backup, specify S3 endpoint with the corresponding S3 region in the format **<S3endpoint>/<S3region>** **apicup subsys set mgmt database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard** • For **sftp** type, the SFTP server hostname **apicup subsys set mgmt database-backup-host=my.sftp.backup.domain.example.com** • Not required for local storage backups
database-backup-path	The path to the location of the backup: • For **objstore** backups, the name of your S3 bucket to store backup data. For example, **bucket-name/folder**. The use of subdirectories in the bucket name is not supported. **apicup subsys set mgmt database-backup-path=apic-backup** Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name. Ensure that **bucket-name/folder** is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error. ◦ **Explanation:** Once a folder is used, archive.info and backup.info files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring. • For **sftp** type, the full absolute path of the folder on the SFTP server, beginning with /. **apicup subsys set mgmt database-backup-path=<*folder_name*>** • Not required for local storage backups
database-backup-port	The port for the protocol to connect to the **database-backup-host**. The backup port is not required for object storage. Default: 22 **apicup subsys set mgmt database-backup-port=<*port_number*>**
database-backup-retries	For **sftp** type backups, the number of times the **ibm-apiconnect** Operator attempts backups in the event of a failed SFTP backup. **apicup subsys set mgmt database-backup-retries=3** Default value: **0**. Not used for **objstore** or local storage backup.

Parameter	Description
`database-backup-schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The backup schedule defaults to `0 0 * * *`. This means a backup is run every day at midnight and minute zero. The timezone for backups is UTC. For local backups, the default is 01:00 UTC. When you configure a host, if you do not specify a value for **`database-backup-schedule`**, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed. `apicup subsys set mgmt database-backup-schedule="0 0 * * *"`

`database-backup-certs`	Custom certificate. Used only when **`s3Provider`** is set to **`custom`**. Most of the custom S3 providers are created based on custom CA certificate.
	
	This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be **`ca.crt`** and value should be base64 encoded value of CA certificate.
	
	The API Connect management subsystem validates the custom S3 certificate against the customer-provided CA certificate.
	
	Sample bash script to create the Kubernetes secret:
	
	`cat >customs3ca.yaml <<EOF`
	`apiVersion: v1`
	`data:`
	`  ca.crt: $(base64 <path-to-ca-certificate> | tr -d '\n' )`
	`kind: Secret`
	`metadata:`
	`  name: custom-server-ca`
	`type: generic`
	`EOF`
	
	`kubectl apply -f customs3ca.yaml -n <namespace>`
	
	For example:
	
	`apicup subsys set mgmt database-backup-certs=my-custom-s3-ca-cert`

`database-backup-s3-uri-style`	Valid values: **`host`** and **`path`**. Default value: **`host`** Only allowed if **`s3Provider`** is set to **`custom`**. Some custom S3 providers require URI style to be set to **`path`**. For example, minIO supports both **`host`** and **`path`** style setup. You can create a minIO S3 server to only accept **`path`** style client communications. **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, upstream custom S3 can reject connections from the client, such as API Connect management subsystem. `apicup subsys set mgmt database-backup-s3-uri-style='<host|path>'`
`database-backup-repo-retention-full`	The number of full S3 or local backups to retain. Not supported for SFTP. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: **`1`** Maximum value: **`9999999`** Default: none (S3 backups), 14 (local backups).

2. Verify that your settings are valid:
   - To view all subsystem settings except **`database-backup-auth-ssh-key`**:

     `apicup subsys get <subsystem_name> --validate`

   - To view **`database-backup-auth-ssh-key`**:

     `apicup certs get <subsystem_name> database-backup-auth-ssh-key --type key`

3. Return to the instructions for installing the management subsystem: Deploying the Management virtual appliance.
   Note that the configuration settings for backups will be installed when you run **`apicup subsys install <subsystem_name>`** after all configuration settings are complete.

4. **S3 backups only**: After you complete the deployment of the Management subsystem in the previous step, continue to Verify configuration for s3 backup V10.0.6.0.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Verify configuration for s3 backup V10.0.6.0

Verify that the configuration for s3 backups succeeded.

About this task

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
When you configure Management subsystem backups, the operator runs a **stanza-create** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

Procedure

1. SSH into the management appliance.
2. Check the status of the **stanza-create** job:

```
kubectl get jobs -n <namespace> | grep stanza
```

- On success:
  - The **stanza-create** job status is **1/1**:

```
kubectl get jobs | grep stanza
m1-b722e361-postgres-stanza-create          1/1           5s         127m
```

  - The **stanza-create** job shows **status: "True"** and **type: Complete**:

```
kubectl get job m1-b722e361-postgres-stanza-create -o yaml

status:
  completionTime: "2021-04-13T18:38:50Z"
  conditions:
    - lastProbeTime: "2021-04-13T18:38:50Z"
      lastTransitionTime: "2021-04-13T18:38:50Z"
      status: "True"
      type: Complete
 startTime: "2021-04-13T18:38:45Z"
 succeeded: 1
```

  - The **stanza-create** pod is in **Completed** state:

```
kubectl get pods | grep stanza
m1-b722e361-postgres-stanza-create-q4fvp                        0/1     Completed   0          127m
```

  - Exec into the pod:

```
kubectl exec -it <backrest-shared-repo-pod> -- bash
```

  - The **pgbackrest** command returns the S3 contents in valid JSON:

```
pgbackrest info --output json --repo1-type s3

"backup":{"held":false}},"message":"ok"}}][pgbackrest@m1-b722e361-postgres-backrest-shared-repo-69cdfdfdd5-rs882
/
```

- On failure:
  - **stanza-create** pods fail, and job status is **0/1**:

```
kubectl get jobs | grep stanza
 m1-3bfe12ac-postgres-stanza-create          0/1           32m        32m
```

  - Listing all pods brought up by **stanza-create** job shows multiple pods because a job tries **backoffLimit** times to complete the job. By default, there will be **7** pods in error state:

```
kubectl get pods | grep stanza
 m1-3bfe12ac-postgres-stanza-create-726z4                       0/1     Error       0          11m
 m1-3bfe12ac-postgres-stanza-create-9vq4n                       0/1     Error       0          22m
 m1-3bfe12ac-postgres-stanza-create-gbtgb                       0/1     Error       0          30m
 m1-3bfe12ac-postgres-stanza-create-gl5g5                       0/1     Error       0          17m
 m1-3bfe12ac-postgres-stanza-create-q8qhw                       0/1     Error       0          26m
 m1-3bfe12ac-postgres-stanza-create-t84zs                       0/1     Error       0          8m54s
 m1-3bfe12ac-postgres-stanza-create-z8tfs                       0/1     Error       0          12m
```

    Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

  - Pod log errors show a reason for the failure. For example, when a hostname is not valid:

```
kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command
requested"
time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest
stanza-create  --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --
repo1-type=s3]"
time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create  --
db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3 "
time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
time="2021-04-13T20:05:21Z" level=info msg="output=[]"
```

  - Job status is set to **type: Failed** and **status: "True"**, with **reason: BackoffLimitExceeded**:

```
kubectl get job m1-3bfe12ac-postgres-stanza-create -o yaml

status:
  conditions:
    - lastProbeTime: "2021-04-13T20:28:03Z"
      lastTransitionTime: "2021-04-13T20:28:03Z"
      message: Job has reached the specified backoff limit
      reason: BackoffLimitExceeded
      status: "True"
      type: Failed
      failed: 7
startTime: "2021-04-13T20:01:18Z"
```

- If pod logs are lost, you can execute a command inside the **backrest-shared-repo** pod:
  - Obtain the **backrest-shared-repo** pod name:

    ```
    kubectl get pods -n <namespace> | grep backrest-shared-repo
    ```

    For example:

    ```
    kubectl get pods | grep backrest-shared-repo
    m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-twl95   1/1     Running
    ```

  - Exec into the pod:

    ```
    kubectl exec -it <backrest-shared-repo-pod> -- bash
    ```

  - Run:

    ```
    pgbackrest info --output json --repo1-type s3
    ```

    For example, for invalid hostname:

    ```
    ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
    ```

    The exact **ERROR:** will vary depending on the misconfiguration. Common errors:

    Invalid access key
    Invalid access key secret
    Invalid bucket region
    Invalid bucket or folder path
- To fix the incorrect settings, see Reconfiguring backup settings for the management subsystem V10.0.6.0.

Reconfiguring backup settings for the management subsystem V10.0.6.0

You can use the **apicup** utility to add or reconfigure backup settings for the management subsystem, after completion of initial installation.

Before you begin

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
If you have a two data center disaster recovery deployment, before you configure S3 backup settings you must first disable 2DCDR on both sites, and then re-enable 2DCDR after reconfiguring your backup settings. Follow these steps:

1. Disable 2DCDR for the management subsystems on both sites:

   ```
   apicup subsys set <management subsystem> multi-site-ha-enabled=false
   apicup subsys install <management subsystem>
   ```

2. Configure your S3 backup settings. You must specify different backup locations for each data center, see Backup and restore considerations for a two data center deployment.
3. Enable 2DCDR for the management subsystems on both sites:

   ```
   apicup subsys set <management subsystem> multi-site-ha-enabled=true
   apicup subsys install <management subsystem>
   ```

About this task

Warning: If you use the S3 backup protocol, changes to the backup settings result in a short period of database downtime.

- After initial installation, use these steps to add or reconfigure the backup settings.
- When you add or reconfigure backups after completion of the initial installation, the steps are the same as configuring during the initial installation, with the addition of using **apicup** to install (activate) the new settings immediately after adjusting the settings. During initial configuration, you typically wait to use **apicup** to activate settings until all other subsystem settings (unrelated to backup) are complete.
- You can add or reconfigure scheduled backups, or generate manual (on-demand) backups.
- When you specify a filepath in a backup setting, be sure to escape any blank spaces in the path.

Note: You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.

Procedure

1. Use **apicup** to specify backup settings:

   ```
   apicup subsys set <subsystem_name> <setting>=<value>
   ```

Table 1. Configuration settings for backup of management subsystem

Parameter	Description

`database-backup-protocol`	The type of the backup. Supported types:
	• `objstore`
	`apicup subsys set mgmt database-backup-protocol=objstore`
	• `sftp`
	`apicup subsys set mgmt database-backup-protocol=sftp`
	The default value is `sftp`.
	• `local` - Local storage backups
	`apicup subsys set mgmt database-backup-protocol=local`
	Note:
	• The public certificate on the S3 storage provider must be signed by a known certificate authority that is trusted by API Connect. Use of an untrusted authority can cause the following error during backup upload: `x509: certificate signed by unknown authority`.

`database-backup-auth-pass`	The password for the server specified in `database-backup-host`. If using object store, this would be the S3 Secret Access Key parameter. The password will be stored in Base64 encoded format. For example: `apicup subsys set mgmt database-backup-auth-pass '<password>'` Note that you cannot use the `=` sign to assign the password to `database-backup-auth-pass`.
`database-backup-auth-user`	The username for the server specified in `database-backup-host`. If using object store, this would be the S3 Secret Key ID. `apicup subsys set mgmt database-backup-auth-user <user_name>`
`database-backup-auth-ssh-key`	**Version 10.0.3.0 or later**: Optional: OpenSSH private key for DB backups using SFTP only. Changing this value will restart the DB. Important: • Use `apicup certs` to set this value. Do not use `apicup subsys`. • Only OpenSSH keys are supported.[1] `apicup certs set mgmt database-backup-auth-ssh-key <ssh private key filepath>`
`database-backup-s3provider`	When using `objstore` type backup, specify the name of the S3 provider to use. Specify one of the supported values: • `ibm` • `aws` • `custom` Not required for local backups or SFTP backups. `apicup subsys set mgmt database-backup-s3provider=ibm`

`database-backup-host`	The backups host:
	• For `objstore` type backup, specify S3 endpoint with the corresponding S3 region in the format `<S3endpoint>/<S3region>`
	`apicup subsys set mgmt database-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud/eu-standard`
	• For `sftp` type, the SFTP server hostname
	`apicup subsys set mgmt database-backup-host=my.sftp.backup.domain.example.com`
	• Not required for local storage backups

`database-backup-path`	The path to the location of the backup:
	• For `objstore` backups, the name of your S3 bucket to store backup data. For example, `bucket-name/folder`. The use of subdirectories in the bucket name is not supported.
	`apicup subsys set mgmt database-backup-path=apic-backup`
	Note: When your deployment includes a management subsystem in two different clusters (with active databases), the two management subsystems cannot use the same s3 bucket name in the database backup configurations. Each management subsystem must use a unique s3 bucket name.
	Ensure that `bucket-name/folder` is empty. If the folder was previously used for backups, the folder will not be empty, and stanza create might encounter an error.
	◦ **Explanation:** Once a folder is used, archive.info and backup.info files are created. During stanza creation, the process compares the database version and database system id between the two info files and the current Postgres database cluster. Stanza creation fails if there is a mismatch. To prevent this possibility, remove all files from the folder before configuring.
	• For `sftp` type, the folder name on the SFTP server
	`apicup subsys set mgmt database-backup-path=<folder_name>`
	• Not required for local storage backups

`database-backup-port`	The port for the protocol to connect to the `database-backup-host`. The backup port is not required for object storage. Default: 22 `apicup subsys set mgmt database-backup-port=<port_number>`
`database-backup-retries`	For `sftp` type backups, the number of times the `ibm-apiconnect` Operator attempts backups in the event of a failed SFTP backup. `apicup subsys set mgmt database-backup-retries=3` Default value: `0`. Not used for `objstore` or local storage backup.

Parameter	Description
`database-backup-schedule`	Cron like schedule for performing automatic backups. The format for the schedule is: • * * * * * • - - - - - • | | | | | • | | | | +----- day of week (0 - 6) (Sunday=0) • | | | +------- month (1 - 12) • | | +--------- day of month (1 - 31) • | +----------- hour (0 - 23) • +------------- min (0 - 59) The backup schedule defaults to `0 0 * * *`. This means a backup is run every day at midnight and minute zero. The timezone for backups is UTC. When you configure a host, if you do not specify a value for **database-backup-schedule**, the default backup schedule is automatically set. Note that the default backup schedule is not set, and scheduled backups not enabled, until host configuration is completed. `apicup subsys set mgmt database-backup-schedule="0 0 * * *"`
`backup-certs`	**Optional**. Backup certificates to use for TLS communication between API Connect and the backup server. Supported for custom s3 backups only. Use this setting when you have set **database-backup-s3provider=custom** For example: `apicup subsys set mgmt backup-certs=<certificate_name>`
`database-backup-s3-uri-style`	**Optional**: Specifies whether the URI is in the form of **host** or **path**. Used only with custom s3providers. Use this setting when you have set **database-backup-s3provider=custom**. `apicup subsys set mgmt database-backup-s3-uri-style='<host|path>'` Supported values: • **host** • **path**
`database-backup-repo-retention-full`	The number of full S3 or local backups to retain. Not supported for SFTP. When the next full backup successfully completes, and the specified number of retained backups is reached, the oldest full backup is deleted. All incremental backups and archives associated with the oldest full backup also expire. Incremental backups are not counted for this setting. Applies to both manual backups and scheduled backups. Minimum value: **1** Maximum value: **9999999** Default: none (S3 backups), 14 (local backups).

Note: To view all subsystem settings for the management subsystem:

```
apicup subsys get <subsystem_name> --validate
```

2. Apply the settings:

   ```
   apicup subsys install <subsystem_name>
   ```

3. Ensure that the changes were applied successfully:

   ```
   apicup subsys health-check <subsystem_name>
   ```

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Backing up the management subsystem V10.0.6.0

Use apicup to create a manual backup of the management subsystem

Before you begin

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
You must have configured backups for the subsystem. See Configuring backup settings during initial installation of the management subsystem V10.0.6.0

About this task

Use this task to create an on-demand, or manual, backup of the management system.

To set up scheduled backups, see the **schedule** option in Configuring backup settings during initial installation of the management subsystem V10.0.6.0.

Procedure

1. Ensure that there are no Management subsystem backups in Running state before creating a new Management backup.
2. Ensure your system is healthy:

   ```
   apicup subsys health-check [subsystem_name]
   ```

When successful, the command returns silently with an exit code of `0`.

3. Make sure that the management cluster status is Running and the READY condition displays the same value before and after the "/".
SSH into the appliance and run `kubectl`:

```
kubectl get mgmt
NAME    READY   STATUS    VERSION      RECONCILED VERSION   AGE
mgmt    16/16   Running   10.0.2.0   10.0.2.0         19h
```

Note: Be sure `READY` status is `16/16`. If you have just configured or reconfigured the backup settings, there is a period of time when the postgres pods are taken offline and brought back up with the new configuration. During this time, the `kubectl get mgmt` command will show `11/11` instead of `16/16`. If you start a backup before the Management cluster is on `16/16`, the backup attempt might stall indefinitely.

4. Create a backup:

```
apicup subsys backup [subsystem_name] [flags]
```

Decide which of the two types of manual backups you want to do:

- `full`: backs up the entire Management subsystem database.

  ```
  apicup subsys backup management --type full
  ```

- `incr`: backs up any data that has not been backed up since the last `full` or `incr` backup.

  ```
  apicup subsys backup management  --type incr
  ```

To view additional flags, use `-h`:

```
./apicup subsys backup management -h
Backup management subsystem
Usage:
  backup SUBSYS [flags]Flags:
      --debug                   Enable debug logging
  -h, --help                    help for backup
      --type string             Type of backup to take ['incr' (backup data starting from last backup), 'full' (complete
backup)] (default "full")
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

Restoring the management subsystem V10.0.6.0

Use `apicup` to list backups and restore a selected backup of the management subsystem

Before you begin

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem

- If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.
- When restoring, the management, analytics, and developer portal subsystems must be at the same version and fix pack level.
- Restoring the Management Service requires database downtime and is a destructive process that deletes current data and copies backup data. During the restoration process, external traffic must be stopped.
- In a Disaster Recovery scenario, do not log in to the administration UI or attempt to configure or change any settings prior to restoring the backup. Restore the backup immediately after installing the subsystem.
- To restore the management database, you must use the original project directory that was created with `apicup` during the initial product installation. You cannot restore the database without the initial project directory because it contains pertinent information about the cluster. The endpoints and certificates cannot change; the same endpoints and certificates will be used in the restored system. Successful restoration depends on use of a *single* `apicup` project for all subsystems, even those in a different cluster. Multiple projects will result in multiple certificate chains which will not match.
- Endpoints for the components cannot change between deployments. However, the endpoints for the VMware hosts can be modified for the new deployment.
- Map the DNS entries from the source cluster to the corresponding IP addresses on the target cluster. Record the DNS entries for each endpoint before starting the restore.
- When restoring the management database, the endpoints (on the new cluster which is the target for the restoration) have to be the same as those on the old cluster (the source of the backup).

About this task

Restoring a backup restores the registration credentials (client_ID, client_secret) that were in use at the time that the selected backup was created. For information on the registration credentials, see Changing the registration client_id and client_secret for applications.

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

Procedure

1. To list the available backups:

```
apicup subsys list-backups [subsystem_name] [flags]
```

Example:

```
# apicup subsys list-backups mgmt
NAME                  STATUS   ID                  CLUSTER      TYPE   CR TYPE   AGE
```

```
management-3aa3bebf   Ready    20200929-152150F   management   full   record    10h
management-3c4ca8df   Ready    20200929-115520F   management   full   record    20h
management-5tbxj      Ready    20200929-224349F   management   full   create    17h
management-f10af337   Ready    20200925-133703F   management   full   record    5d2h
management-jtlcd      Ready    20200929-224349F   management   full   create    25h
management-zxjtz      Ready    20200929-224349F   management   full   create    28h
```

Optional flags:

```
./apicup subsys list-backups -h
List backups of the subsystem
Usage:
  apicup subsys list-backups SUBSYS [flags]
Flags:
  -h, --help               help for list-backups
      --timeout duration   Command timeout in seconds. (default 40s)
Global Flags:
      --accept-license   Accept the license for API Connect
      --debug            Enable debug logging
```

Attention: You can restore the Management subsystem from any backup where the STATUS is Ready; however, you should avoid restoring to the initial system backup. During installation, the Management database is used for an initial system backup before certain database schema jobs are complete. Restoring to this backup will result in an unstable system.

2. To restore a selected backup:

```
apicup subsys restore [subsystem_name] --name <backup_name> [flags]
```

Example, where **management-zxjtz** is the value of **<backup_name>**, chosen from the backup **NAMES** listed by **apicup subsys list-backups** in Step 1:

```
apicup subsys restore mgmt --name management-zxjtz
```

Optional flags:

```
apicup subsys restore management -h
Restore management subsystem

Usage:
  restore SUBSYS_NAME [flags]

Flags:
      --debug                                Enable debug logging
  -h, --help                                 help for restore
      --name string                          Name of the backup to restore
      --wait                                 Wait for the operation to complete or fail.
      --wait-timeout duration                Command timeout in seconds. (default 40s)
```

3. You can list your restores and their statuses.

```
apicup subsys list-restores [subsystem_name] [flags]
```

To view the optional flags, use **-h**:

```
apicup subsys list-restores -h
List restores of the subsystem

Usage:
  apicup subsys list-restores SUBSYS [flags]

Flags:
  -h, --help               help for list-restores
      --timeout duration   Command timeout in seconds. (default 40s)

Global Flags:
      --accept-license   Accept the license for API Connect
      --debug            Enable debug logging
```

Troubleshooting management subsystem backups on VMware V10.0.6.0

You can troubleshoot failures of the management subsystem backups.

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- **Troubleshooting resiliency issues V10.0.6.0**
  Resolve resiliency issues with API Connect management system backups.
- **Troubleshooting stanza-create job for S3 backup configuration V10.0.6.0**
  Troubleshoot configuration of s3 backups when the stanza-create job fails.
- **Restore failure with compliance pod error on VMware V10.0.6.0**
  Compliance pod reports an error, and restore does not progress.

Troubleshooting resiliency issues V10.0.6.0

Resolve resiliency issues with API Connect management system backups.

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem

pgbackrest-shared-repo resiliency problems

If the Kubernetes node hosting pgbackrest-shared-repo pod is down and the PVC attached to the pod has strict zone requirements (for example, in AWS or other clouds) or if the storage class is set to **local-storage**, then pgbackrest-shared-repo pod will not get rescheduled to another Kubernetes node.

As a result, there is a single point of failure and the following conditions might occur:

- Backups of the management database fails
- Disk space fills with accumulated Postgres wal (Write-Ahead Logging) files

To avoid this problem, monitor the disk usage: Monitoring Postgres disk usage on VMware.

When the Kubernetes node comes back up, the pod is scheduled and all the processes should resume properly.

Troubleshooting stanza-create job for S3 backup configuration V10.0.6.0

Troubleshoot configuration of s3 backups when the stanza-create job fails.

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
When you configure Management subsystem backups for s3 providers, the operator runs a **stanza-create** job. This job creates the stanza (Postgres cluster configuration) on the upstream S3 server, which is used for backup and archive procedures. The job also brings up the necessary pod.

SSH into the appliance VM, and check the status of the **stanza-create** job:

```
kubectl get jobs -n <namespace> | grep stanza
```

On failure:

- **stanza-create** pods fail, and job status is **0/1**:

  ```
  kubectl get jobs | grep stanza
   m1-3bfe12ac-postgres-stanza-create          0/1           32m        32m
  ```

- Listing all pods brought up by **stanza-create** job shows multiple pods because a job tries **backoffLimit** times to complete the job. By default, there will be **7** pods in error state:

  ```
  kubectl get pods | grep stanza
   m1-3bfe12ac-postgres-stanza-create-726z4                      0/1     Error       0          11m
   m1-3bfe12ac-postgres-stanza-create-9vq4n                      0/1     Error       0          22m
   m1-3bfe12ac-postgres-stanza-create-gbtgb                      0/1     Error       0          30m
   m1-3bfe12ac-postgres-stanza-create-gl5g5                      0/1     Error       0          17m
   m1-3bfe12ac-postgres-stanza-create-q8qhw                      0/1     Error       0          26m
   m1-3bfe12ac-postgres-stanza-create-t84zs                      0/1     Error       0          8m54s
   m1-3bfe12ac-postgres-stanza-create-z8tfs                      0/1     Error       0          12m
  ```

  Note: In some cases, the pods are cleaned by Kubernetes and the logs of these pods can be lost.

- Pod log errors show a reason for the failure. For example, when a hostname is not valid:

  ```
  kubectl logs m1-3bfe12ac-postgres-stanza-create-gbtgb

  time="2021-04-13T20:01:19Z" level=info msg="pgo-backrest starts"
  time="2021-04-13T20:01:19Z" level=info msg="debug flag set to false"
  time="2021-04-13T20:01:19Z" level=info msg="backrest stanza-create command
  requested"
  time="2021-04-13T20:01:19Z" level=info msg="s3 flag enabled for backrest command"
  time="2021-04-13T20:01:19Z" level=info msg="command to execute is [pgbackrest
  stanza-create  --db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --
  repo1-type=s3]"
  time="2021-04-13T20:01:19Z" level=info msg="command is pgbackrest stanza-create  --
  db-host=192.1.2.3 --db-path=/pgdata/m1-3bfe12ac-postgres --repo1-type=s3 "
  time="2021-04-13T20:05:21Z" level=error msg="command terminated with exit code 49"
  time="2021-04-13T20:05:21Z" level=info msg="output=[]"
  ```

- Job status is set to **type: Failed** and **status: "True"**, with **reason: BackoffLimitExceeded**:

  ```
   kubectl get job m1-3bfe12ac-postgres-stanza-create -o yaml

  status:
    conditions:
      - lastProbeTime: "2021-04-13T20:28:03Z"
        lastTransitionTime: "2021-04-13T20:28:03Z"
        message: Job has reached the specified backoff limit
        reason: BackoffLimitExceeded
        status: "True"
        type: Failed
        failed: 7
  startTime: "2021-04-13T20:01:18Z"
  ```

- If pod logs are lost, you can execute a command inside the **backrest-shared-repo** pod:
  1. Obtain the **backrest-shared-repo** pod name:

     ```
     kubectl get pods -n <namespace> | grep backrest-shared-repo
     ```

For example:

```
kubectl get pods | grep backrest-shared-repo
m1-eb8edc18-postgres-backrest-shared-repo-98dd46cc6-twl95   1/1     Running
```

2. Exec into the pod:

   ```
   kubectl exec -it <backrest-shared-repo-pod> -- bash
   ```

3. Run:

   ```
   pgbackrest info --output json --repo1-type s3
   ```

   For example, for invalid hostname:

   ```
   ERROR: [049]: unable to get address for 'test1-v10-backup.s3.exampleaws.com': [-2] Name or service not knownCopy
   ```

   The exact `ERROR:` will vary depending on the misconfiguration. Common errors:

   Invalid access key
   Invalid access key secret
   Invalid bucket region
   Invalid bucket or folder path

- To fix the incorrect settings, see Reconfiguring backup settings for the management subsystem V10.0.6.0.

Restore failure with compliance pod error on VMware V10.0.6.0

Compliance pod reports an error, and restore does not progress.

Note: This page is for v10.0.6 only. For v10.0.7 the management backup documentation is Backing up and restoring the Management subsystem
This failure can happen in a situation where a backup is taken on a system where API governance has never been enabled, and the restore of that backup is done onto a system which has API governance enabled. In this situation the API governance database is missing, and consequently the compliance pods report an error.

Symptoms

The compliance service pods are continually restarting, and the restore does not progress. You can SSH into the appliance VM, and check the log of the compliance pod. For example:

```
kubectl logs management-up-compliance-data-populate-0-to-<nnn>-<xx>-<yy>
```

The log of the compliance pod shows an error like the following example:

```
2023-05-19T03:34:31.771Z bhendi:error Database availability check to management-0bfd2507-postgres:5432 failed, try again in
2000 ms,
error: : database "compliance" does not exist, stack: error: database "compliance" does not exist
```

Resolution

Delete all of the compliance jobs, and then start the restore again. For example:

```
kubectl -n <namespace> delete job management-up-compliance-service-data-populate-0-to-1 management-up-compliance-service-
schema-0-to-1
```

Backing up and restoring the Developer Portal

You can backup and restore a Developer Portal in a VMware environment.

About this task

It is strongly recommended that you configure the backup parameters for your portal service during installation. If you did not do so when you installed API Connect in your runtime environment, you must configure the backups for the Developer Portal before performing an upgrade. These backups can then be used to restore the Developer Portal if required. When your Developer Portal subsystem is running, you can also make on-demand backups by using the command line.

Note:

- You must backup both the management and portal subsystems at the same time, to ensure synchronicity across the services.
- When restoring, the management, analytics, and developer portal subsystems must be at the same version and fix pack level.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

- To complete a backup and restore of your Portal Subsystem, you must use apicup to configure the backup settings.
- Backups can be scheduled or generated manually (on-demand).

- You can then list the available backups are available and select one to restore your Portal Subsystem when necessary.
- Supported backup types:
  - Cloud-based S3 storage
    - IBM Cloud Object Storage
    - AWS S3
  - SFTP

- **Configuring backup settings for the Developer Portal subsystem**
  You can use the **apicup** utility to configure backup settings for the Developer Portal subsystem
- **Backing up the Developer Portal subsystem**
  After you have configured backup settings, you can use **apicup** to create a manual backup of the Developer Portal.
- **Restoring the Developer Portal subsystem**
  You can use **apicup** to restore the Developer Portal subsystem.

Configuring backup settings for the Developer Portal subsystem

You can use the **apicup** utility to configure backup settings for the Developer Portal subsystem

Before you begin

Review Backing up and restoring the Developer Portal.

About this task

- During an initial installation, use these steps to configure the backup settings prior to creating the initial management ISO.
- You can configure scheduled backups, or generate manual (on-demand) backups.
- OpenSSH key authentication for SFTP is supported.
- When you specify a filepath in a backup setting, be sure to escape any blank spaces in the path.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

Tip: If you are using remote backups, and there are a large number of rows in your **portalbackup** list that have a **CR TYPE** of **record**, there might be a slow down in system performance, including during Developer Portal upgrades. Although there is no upper limit to how many portal backups you can have, it is recommended that you prune your remote backups so that the number of rows does not become excessively large. One **portalbackup** row of type **record**, corresponds to one backup file on the configured remote backup server. For more information, see Overview of the Developer Portal backup resources v10.0.6.0.

Procedure

1. Open your API Connect installation project directory.
2. Run the following commands to set the location and timing of your backups:

```
apicup subsys set <subsystem_name> <setting>=<value>

apicup subsys set ptl site-backup-host=mybackuphost.com
apicup subsys set ptl site-backup-port=22
apicup subsys set ptl site-backup-auth-user=mybackupauthusername
apicup subsys set ptl site-backup-auth-pass=mybackupauthpassword
apicup subsys set ptl site-backup-path=/site-backups
apicup subsys set ptl site-backup-protocol=sftp
apicup subsys set ptl site-backup-certs=<custom-s3-server-CA-cert>
apicup subsys set ptl site-backup-s3-uri-style=<host-or-path>
apicup subsys set ptl site-backup-schedule="0 2 * * *"
apicup subsys set ptl site-priority-list=portal.a1.example.com/test-1/spacecatalog,portal.a1.example.com/test-
4/cat3,portal.a1.example.com/test-3/cat1
```

Use **apicup certs** instead of **apicup subsys** to configure ssh-key authentication for SFTP:

```
apicup certs set ptl site-backup-auth-ssh-key <ssh-key-file-path>
```

The backup parameters are detailed in the following table.

Table 1. Portal backup parameters

Parameter	Description
site-backup-host	The fully qualified domain name of the backup server, in lowercase only. • For **objstore** type backup, specify the S3 endpoint with the corresponding S3 region in the format **<S3endpoint>**, or with an optional S3 region in the format **<S3endpoint>/<S3region>**. **apicup subsys set ptl site-backup-host=s3.eu-gb.cloud-object-storage.appdomain.cloud** • For **sftp** type, the SFTP server hostname **apicup subsys set ptl site-backup-host=my.sftp.backup.domain.example.com**

Parameter	Description
site-backup-port	The port for the protocol to connect to the **site-backup-host**. Defaults to **22** if not explicitly set. The backup port is not required for object storage but if specified, is typically port 443. **apicup subsys set ptl site-backup-port=22**
site-backup-auth-user	The user name for the server specified in **site-backup-host**. If using object storage, the user name is the S3 Secret Key ID. **apicup subsys set ptl site-backup-auth-user=<*user_name*>**
site-backup-auth-pass	The password for the server specified in **site-backup-host**. If using object storage, the password is the S3 Secret Access Key parameter. The password is stored in Base64 encoded format, and must not be edited directly in the apiconnect-up.yml file. **apicup subsys set ptl site-backup-auth-pass=*password***
site-backup-auth-ssh-key	**Version 10.0.3.0 or later:** OpenSSH key for database backups by using SFTP. Important: • Use **apicup certs** to set this value. Do not use **apicup subsys**. • Only OpenSSH keys are supported.[1] **apicup certs set ptl site-backup-auth-ssh-key=<*ssh-key-file-path*>**
site-backup-path	The location of where the backups are stored: • S3 - the name of your S3 bucket to store backup data, optionally followed by a **/** and then any subdirectories inside the bucket. Valid examples: **bucket** **bucket/path** **bucket/lots/of/paths** Example: **apicup subsys set ptl site-backup-path=apic-backup** • SFTP - the full absolute path of the folder on the SFTP server, beginning with **/**. **apicup subsys set ptl site-backup-path=<*/folder*>**
site-backup-protocol	The protocol that is used to communicate with your remote backup endpoint. Specify one of the following values: • **sftp** - for secure file transfer protocol • **local** - for local storage • **objstore** - for S3 compatible object storage Starting with v10.0.3.0, the default protocol is **local** (earlier releases used **sftp** as the default). For example: **apicup subsys set ptl site-backup-protocol=local**
site-backup-certs	The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be **ca.crt** and the value is the base64-encoded value of the CA certificate. If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate. If you do need to provide the CA certificate then you must provide the entire chain in the **ca.crt** member of the secret, as follows: **intermediate-certificate-1** **intermediate-certificate-2** **. . .** **intermediate-certificate-*n*** **root-certificate** Where the first certificate in the **ca.crt** member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the **ca.crt** member is the issuer of the preceding certificate. The root certificate in the **ca.crt** member must be self-signed. If there are no intermediate certificates involved, then the **ca.crt** member contains only the root certificate. The following sample bash script creates a Kubernetes secret: **cat >customs3ca.yaml <<EOF** **apiVersion: v1** **data:** **ca.crt: $(base64 <*path-to-ca-certificate*> | tr -d '\n')** **kind: Secret** **metadata:** **name: custom-server-ca** **type: generic** **EOF** **kubectl apply -f customs3ca.yaml -n <*namespace*>**
site-backup-s3-uri-style	Optional. Indicates whether URIs to your S3 backup should use specify a host or a path value. When not specified, defaults to **host**. Supported beginning with 10.0.2. Valid values: **host** and **path**. Some custom S3 providers require URI style to be set to **path**. For example, MinIO supports both **host** and **path** style setup. You can create a MinIO S3 server to only accept **path** style client communications. **Important**: Contact your custom S3 administrator before configuring this field. If not properly configured, an upstream custom S3 can reject connections from the client, such as the API Connect Management subsystem.

Parameter	Description

`site-backup-schedule`	The schedule for how often automatic Portal backups are run. The format for the schedule is any valid cron string, as follows:
	
	`* * * * *`
	`- - - - -`
	`| | | | |`
	`| | | | +----- day of week (0 - 6) (Sunday=0)`
	`| | | +------- month (1 - 12)`
	`| | +--------- day of month (1 - 31)`
	`| +----------- hour (0 - 23)`
	`+------------- min (0 - 59)`
	
	For example: `30 22 * * 1` performs backups at 10:30 pm on Mondays.
	The default backup schedule is `0 2 * * *` (runs every day at 2 am).
	
	The timezone for backups is UTC.
	
	`apicup subsys set ptl site-backup-schedule="0 2 * * *"`

`site-priority-list`	Optional list of prioritized sites to restore. Default: empty.
	
	`apicup subsys set <portal-subsystem-name> site-priority-list <comma-separated-list-of-sites>`
	
	For example:
	
	`apicup subsys set ptl site-priority-list portal.a1.example.com/e2etest-1/spacecatalog,portal.a1.example.com/e2etest-4/cat3,portal.a1.example.com/e2etest-3/cat1`

3. At any time you can view the current Portal subsystem values:
   - For most values:

     `apicup subsys get ptl`

     where ***ptl*** is the name that you assigned to your Portal service. The output from this command lists all of the subsystem settings, including backup, and indicates whether there are any errors. You must fix any errors before continuing.

   - For ssh-key authentication for SFTP:

     `apicup certs get ptl site-backup-auth-ssh-key`

4. Activate the backup settings by running the following command:

   `apicup subsys install ptl`

   where ***ptl*** is the name that you assigned to your Portal service.
5. Validate the installation with the new backup parameters by running the following command:

   `apicup subsys get ptl --validate`

   where ***ptl*** is the name that you assigned to your Portal service. The output from this command lists all of the subsystem settings, including backup, and indicates whether the values are valid or invalid. You must correct any invalid values before continuing.
6. Ensure that the changes were applied successfully:

   `apicup subsys health-check <subsystem_name>`

- **Overview of the Developer Portal backup resources v10.0.6.0**
  How to manage your Developer Portal remote backup resources.

[1] PuTTY style keys can be converted to OpenSSH by using the PuTTY Key Generator (PuTTYgen) application; see https://www.puttygen.com/.

Overview of the Developer Portal backup resources v10.0.6.0

How to manage your Developer Portal remote backup resources.

Note: This page is for V10.0.6.0 only. For V10.0.7.0 the backup, restore, and disaster recovery processes are documented in Backing up, restoring, and disaster recovery.
Note: This information is relevant only if you're taking remote Developer Portal backups.
The Developer Portal uses two types of portal backup resources. When you run the command **`kubectl get portalbackup`**, you see a list of backups, for example:

```
NAME                  ID                TASK ID      STATUS   TYPE     CR TYPE   CLUSTER   AGE     COMMENT
my-backup-f91ms       n/a               20b47fe090f5 Ready    system   create    portal    78m
portal-record-222cg   20220108.010033                Ready    system   record    portal    94m
portal-record-22xdc   20220702.010339                Ready    site     record    portal    9m49s
portal-record-245j7   20221002.010340                Ready    site     record    portal    5m
portal-record-246hm   20221218.010103                Ready    site     record    portal    32m
portal-record-24h44   20220509.010037                Ready    site     record    portal    68m
portal-record-24wb4   20221003.010029                Ready    site     record    portal    60m
portal-record-264mh   20221119.010453                Ready    system   record    portal    78m
```

There are two types of resources listed, and these are denoted by the **`CR TYPE`** column; **`create`** or **`record`** types.

1. **`create`**: This resource is a representation of a **`portalbackup`** task that was created manually. See Backing up and restoring the Developer Portal in a Kubernetes environment v10.0.6.0, Backing up Developer Portal on OpenShift and Cloud Pak for Integration v10.0.6.0, or Backing up the Developer Portal subsystem on VMware, for information about how to create a manual portal backup.
2. **`record`**: This resource is generated by the operator, and it's a representation of a backup file that's stored on the remote backup server. A record listing can be modified only to add a comment to it. You can describe each record to understand which file it is referring to by running the following command:

   `kubectl describe portalbackup portal-record-name`

For example:

```
kubectl describe portalbackup portal-record-zmmfw
.
.
Name:         portal-record-zmmfw
Namespace:    apic
Labels:       createdBy
.
.
Full Backup Name:        my-portal-s3-hostname@my-porg@catal-20220604.010302.tar.gz
```

Record rows are generated and updated on a schedule, according to the backups that exist on your remote backup server. If you have 10 backup files stored on your remote server, then there will be 10 record rows; one row for each file.

The Developer Portal doesn't have a retention policy for remote backups. Therefore, it's your responsibility to ensure that you're keeping the number of backups that are stored remotely under control. An excessive number of portal backups results in delays administrating and upgrading the Developer Portal. This is due to the operator needing to maintain the excessive number of record rows that correspond to each remote portal backup.

Backing up the Developer Portal subsystem

After you have configured backup settings, you can use **apicup** to create a manual backup of the Developer Portal.

About this task

You can make on-demand backups of your Developer Portal system, sites, and complete service, by running the following **apicup** commands in your API Connect installation project directory.

The default Developer Portal backup schedule is once every 24 hours, but the schedule can be changed in the backup settings. The Developer Portal saves all system and site backups locally, and also saves them remotely based on the configured SFTP and s3 settings.

The local backups are automatically maintained so that the latest three backups of each site and of the system are kept, and older backups are removed. This maintenance means that the Developer Portal retains the latest three backups for each site and for the system however old they are, but there is no deletion of the old backups on the remote server. If a site is deleted, then all of the local backups for that site are also deleted, as otherwise the backup volume might become full of old site backups. For remote backups, you can configure a retention policy on your remote server to remove the old backup files as required.

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

Procedure

1. Ensure your system is healthy:

   ```
   apicup subsys health-check [subsystem_name]
   ```

   When successful, the command returns silently with an exit code of **0**.

2. Create a backup:

   ```
   apicup subsys backup [subsystem_name] [flags]
   ```

   - To backup the Portal system configuration (and not the sites), run the following command:

     ```
     apicup subsys backup ptl --type system
     ```

   - To backup a specific Portal site or all sites, run the following command:

     ```
     apicup subsys backup ptl --type site --site-name arg
     ```

     where *arg* is a specific site UUID or URL, or to backup all sites use the argument **installed**.
   - To backup the entire Portal service (the system and all installed sites), run the following command:

     ```
     apicup subsys backup ptl
     ```

     Where *ptl* is the name that you assigned to your Portal service.
     Note: To restore a Developer Portal service, you need backups of both the Portal system and the installed sites.

   To view additional flags, use **-h**:

```
apicup subsys backup portal -h
Backup portal subsystem

Usage:
  backup SUBSYS [flags]

Flags:
      --comment string          Comment to add to the backup CR before creation
      --debug                   Enable debug logging
  -h, --help                    help for backup
      --site-name string        Site to restore to the remote server if TYPE=site. Accepts site uuid/url or 'installed' to
backup all installed sites.
      --type string             Type of backup to take ['system' (backup Portal System), 'site' (backup Portal Site),
'all' (complete backup)] (default "all")
```

```
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

3. To view the available backups and their status:

   ```
   apicup subsys list-backups [subsystem_name] [flags]
   ```

   Optional flags:

   ```
   ./apicup subsys list-backups -h
   List backups of the subsystem
   Usage:
     apicup subsys list-backups SUBSYS [flags]
   Flags:
     -h, --help               help for list-backups
         --timeout duration   Command timeout in seconds. (default 40s)
   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

Restoring the Developer Portal subsystem

You can use **apicup** to restore the Developer Portal subsystem.

About this task

You can restore your Developer Portal service by using the backups that exist on your remote server, by running the following **apicup** commands in your API Connect installation project directory. Note that before you can run these commands, you must have configured your remote backup server details, and have valid backups of both the Portal system and the installed sites. See Configuring backup settings for the Developer Portal subsystem and Backing up the Developer Portal subsystem.
Note:

- If you have to perform a restore, you must complete the restoration of the management service first, and then immediately restore the developer portal. The backups of the management and portal must be taken at the same time to ensure that the portal sites are consistent with management database.
- Restoration requires a functioning Developer Portal. In a disaster recovery scenario, you might need to reinstall the Developer Portal subsystem before you can restore the backed-up data. For disaster recovery, see Recovering the Developer Portal subsystem on VMware.

Procedure

1. To view the available backups and their status:

   ```
   apicup subsys list-backups subsystem_name [flags]
   ```

   Optional flags:

   ```
   ./apicup subsys list-backups -h
   List backups of the subsystem
   Usage:
     apicup subsys list-backups SUBSYS [flags]
   Flags:
     -h, --help               help for list-backups
         --timeout duration   Command timeout in seconds. (default 40s)
   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

2. Review which restore actions will be performed
   To see what restore actions are performed when the **apicup subsys restore** *ptl* command is run, you can use the following command:

   ```
   apicup subsys restore ptl --dry-run --timestamp [now|YYYYMMDD.HHMMSS]
   ```

   where **--timestamp** can be **now**, so the latest backup file that is available is used. Or you can specify a timestamp in the format of ***YYYYMMDD.HHMMSS*** to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given.
   Note:
   - From IBM® API Connect Version 10.0.3.0, the timestamp format changed to ***YYYYMMDD.HHMMSS***. For Version 10.0.2.0 and earlier, the timestamp format is ***YYYY-MM-DD HH:MM:SS***.
   - With the **--dry-run** option, the utility runs through the restore steps without executing them. You can use this option to see if the restore from the selected backup will succeed.
     The **--dry-run** option starts with the following output message:

     ```
     Starting restore...please check later for completion status of restore
     PortalRestore is Running
     ```

     To view the results of the dry-run, you can either look in the portal restore CR for a message of completion (or failure) or you can get a high-level status by entering:

     ```
     apicup subsys list-restores <subsys_name>
     ```

     Status values can be: **Ready**, **Completed**, **Failed** or **Running**.
3. Run the following command, and attach any flags as appropriate.

   ```
   apicup subsys restore [SUBSYS_NAME] [flags]
   ```

   - To restore your Portal service, run the following command:

```
apicup subsys restore ptl --timestamp [now|YYYYMMDD.HHMMSS]
```

where `--timestamp` can be `now`, so the latest backup file that is available is used. Or you can specify a timestamp in the format of `YYYYMMDD.HHMMSS` to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given. This command executes the portal restore process, which downloads the system backup and all of the site backups from the remote server, and installs them within the Portal stack. This process will then restore the system configuration from the found backup, and restore all of the sites. Note that if a backed up site is already installed on the current stack, then the site is reinstalled by using the backup (the site is overwritten by the backup). If there are multiple sites to restore, then these sites are queued for restoring. You can track the restoration process within the Portal `www admin` logs.

- By default the restore is a complete restore. You can optionally include `--type all` on the command line to specify a complete restore, but the flag is not required, as a complete restore is performed by default. Note that a complete restore supports only remote restores.
  Note that the complete restore uses the `priorityList` configuration setting that you specified in Configuring backup settings for the Developer Portal subsystem.

- To restore just the Portal system configuration, run the following command:

  ```
  apicup subsys restore ptl --type system --system-name arg
  ```

  where `arg` can be a `.tgz` name or also accepts `backup ID` if `TYPE=system` and `CR TYPE=record` from the `list-backups` command.

  Note that if Portal system configuration information exists on the current stack, running this command will overwrite that configuration.

- To restore just the Portal sites, run the following command:

  ```
  apicup subsys restore ptl --type site --site-name arg
  ```

  where `arg` can be the site backup tgz file name or URL, to restore a particular site by using the specified backup ID (or the latest backup file found if using a URL). Or you can use the argument `all` to restore all of the Portal sites that have backup files on the remote server. Note that if any of the backed up sites are already installed on the current stack, then they are reinstalled by using the backup (the sites are overwritten by the backup).

```
❯ ./apicup subsys restore <portal> -h
Restore portal subsystem

Usage:
  restore SUBSYS [flags]

Flags:
      --custom-platform-api-hostname string   The custom platform API hostname to override for the portal system when
restoring it. Used only when performing a portal restore system or portal restore all.
      --custom-url string                     The custom URL to override for the portal site when restoring it. Used only
when performing a portal site restore.
      --debug                                 Enable debug logging
      --dry-run                               Runs a dry run of the command, returning the actions that will be taken.
  -h, --help                                  help for restore
      --site-name string                      File to restore site from which should exist on the pod's local filesystem
or the remote server. Also accepts backup ID if TYPE=site and CR TYPE=record.
      --system-name string                    File to restore system from which should exist on the pod's local filesystem
or the remote server. Also accepts backup ID if TYPE=system and CR TYPE=record.
      --timestamp string                      In YYYYMMDD.HHMMSS format, specify a timestamp to retrieve the backup from.
The nearest backup, searching backwards from this timestamp, is used.
      --type string                           Type of restore to take ['system' (restore Portal System), 'site' (restore
Portal Site), 'all' (complete restore)] (default "all")
      --wait                                  Wait for the operation to complete or fail.
      --wait-timeout duration                 Command timeout in seconds. (default 40s
```

4. To view your restores and their statuses, run:

   ```
   apicup subsys list-restores <subsys_name> <flags>
   ```

   Help output:

   ```
   apicup subsys list-restores -h
   List restores of the subsystem

   Usage:
     apicup subsys list-restores SUBSYS [flags]

   Flags:
     -h, --help               help for list-restores
         --timeout duration   Command timeout in seconds. (default 40s)

   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

5. To list the Portal backup sites, use the Portal CLI command. See Using the **sites** commands.
6. To list the Portal system sites, use the Portal CLI command. See Using the **platforms** commands.

Backing up and restoring the Analytics database on VMware

The IBM® API Connect Analytics database can be backed up and restored from an S3 repository. S3 compatible object storage is required; for example, IBM Cloud Object Storage.

About this task

Begin by configuring your backup configuration, which optionally includes a schedule if you want to automate your backups. You can run backups manually as needed. To restore a backup, review the set of available backups, and then restore the selected backup.

- **Configuring backup settings for Analytics**
  Use the `apicup` utility to configure backup settings for the Analytics database in your VMware environment.
- **Running a backup of the Analytics database**
  Initiate a backup of the API Connect Analytics database on VMware.
- **Restoring the Analytics database**
  Restore a selected backup of the IBM API Connect Analytics database on VMware.

Configuring backup settings for Analytics

Use the `apicup` utility to configure backup settings for the Analytics database in your VMware environment.

About this task

The Analytics database can be backed up and restored from an S3 repository. S3-compatible object storage is required; for example, IBM Cloud Object Storage.
Note:

- Only host-style S3 backups are supported. Path-style S3 backups are not supported.
- If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.
- When you specify a filepath in a backup setting, be sure to escape any blank spaces in the path.

Procedure

1. Use the `apicup` utility to configure the backup settings.
   Configure each setting by running the following command with the key and value from Table 1 that you want to apply:

   ```
   apicup subsys set <analytics_subsystem> <key> <value>
   ```

   Table 1 shows the supported backup settings with the default value (if any) and a description of each setting.

   Table 1. Analytics backup settings

Key	Default value	Description
`analytics-backup-auth-pass`		Server password for Analytics backups.
`analytics-backup-auth-user`		Server user name for Analytics backups.

`analytics-backup-host`		The S3 endpoint with the corresponding S3 region in the format:
		`s3.s3region.s3domain`
		For example:
		`s3.eu-gb.cloud-object-storage.appdomain.cloud`
		Attention: A path style host value (formatted as `S3endpoint/S3region`) is not supported for AWS.

Key	Default value	Description

`analytics-backup-certs`		The name of a custom certificate that contains your upstream custom S3 CA certificate. Supported beginning with 10.0.2. This field accepts name of the Kubernetes secret containing your upstream custom S3 CA certificate. The key of the secret must be `ca.crt` and the value is the base64-encoded value of the CA certificate.
		
		If the server certificate that is presented by your S3/object-store endpoint is signed by a well known CA and includes any intermediate certificates in the chain, then you do not need to provide the CA certificate using this feature because the Portal subsystem already trusts the server certificate. You can test this by configuring the Portal backup to your S3/object-store server without providing the CA certificate.
		
		If you do need to provide the CA certificate then you must provide the entire chain in the `ca.crt` member of the secret, as follows:
		
		`intermediate-certificate-1`
		`intermediate-certificate-2`
		`. . .`
		`intermediate-certificate-n`
		`root-certificate`
		
		Where the first certificate in the `ca.crt` member (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate in the `ca.crt` member is the issuer of the preceding certificate. The root certificate in the `ca.crt` member must be self-signed.
		
		If there are no intermediate certificates involved, then the `ca.crt` member contains only the root certificate.
		
		The following sample bash script creates a Kubernetes secret:
		
		`cat >customs3ca.yaml <<EOF`
		`apiVersion: v1`
		`data:`
		`  ca.crt: $(base64 <path-to-ca-certificate> | tr -d '\n' )`
		`kind: Secret`
		`metadata:`
		`  name: custom-server-ca`
		`type: generic`
		`EOF`
		
		`kubectl apply -f customs3ca.yaml -n <namespace>`

`analytics-backup-path`		A combination of the S3 bucket and the base path within the bucket, using the format: `bucket_name/path`
`analytics-backup-chunk-size`	1GB	(Optional) Specifies how large files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks as GB, MB, or KB.
`analytics-backup-schedule`	0 2 * * *	(Optional) Determines how often backups will be invoked automatically. The format for the schedule is any valid `cron` string. The timezone for backups is UTC. Backing up once a day is probably sufficient; remember that the more frequently you run backups, the more storage space you need.
`analytics-enable-compression`	true	(Optional) Determines whether metadata files are stored in compressed format.
`analytics-enable-server-side-encryption`	false	(Optional) Determines whether files are encrypted. When set to true, files are encrypted on the server side using AES256.

2. Validate the backup settings by running the following command:

   `apicup subsys get <analytics_subsystem> --validate`

3. Apply the backup settings to the Analytics subsystem by reinstalling with the following command:

   `apicup subsys install <analytics_subsystem>`

4. Verify that the backup settings were successfully applied by running the following command:

   `apicup subsys health-check <analytics_subsystem>`

Running a backup of the Analytics database

Initiate a backup of the API Connect Analytics database on VMware.

About this task

Procedure

1. Perform a health check of your Analytics system by running the following command:

   `apicup subsys health-check <analytics_subsystem>`

   If successful, the command returns silently with exit code 0.

2. Start the backup by running the following command and including the appropriate flags:

   `apicup subsys backup <analytics_subsystem>`

   View the list of supported flags by running the `backup` command with the `-h` (help) flag, as shown in the following example:

```
apicup subsys backup analytics -h
Backup analytics subsystem

Usage:
  backup SUBSYS [flags]

Flags:
      --comment string          Comment to add to the backup CR before creation
      --debug                   Enable debug logging
  -h, --help                    help for backup
      --ignore-unavailable      If false, the backup will fail if an index is missing. If true, missing indices will be
skipped and the backup will continue.
      --indices strings         Indice(s) you'd like to backup (all/apievents/ui/summary). It can be set to any
combination of indices (e.g. ui or ui,apievents).
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

Note: The `--indices` flag can be set to `all|apievents|ui|summary`, or the names of individual indexes. `all` includes `apievents`, `ui`, and `summary` indices. Multiple values can be supplied with a comma-separated list. For example, to backup all UI data and all data from August of 2022:

```
--indices ui, "apic-api-2022.08*"
```

3. Optionally view a list of available backups, and the status of each, by running the following command:

   ```
   apicup subsys list-backups <analytics_subsystem>
   ```

   View the list of supported flags by running the `list-backups` command with the `-h` (help) flag, as shown in the following example:

   ```
   apicup subsys list-backups -h
   List backups of the subsystem

   Usage:
     apicup subsys list-backups SUBSYS [flags]

   Flags:
     -h, --help               help for list-backups
         --timeout duration   Command timeout in seconds. (default 40s)

   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

Restoring the Analytics database

Restore a selected backup of the IBM® API Connect Analytics database on VMware.

About this task

Use the `apicup` utility to view the list of available backups for the Analytics database, and then restore a selected backup.

Procedure

1. Retrieve a list of available backups, and the status of each, by running the following command:

   ```
   apicup subsys list-backups <analytics_subsystem>
   ```

   View the list of supported flags by running the `list-backups` command with the `-h` (help) flag, as shown in the following example:

   ```
   apicup subsys list-backups -h
   List backups of the subsystem

   Usage:
     apicup subsys list-backups SUBSYS [flags]

   Flags:
     -h, --help               help for list-backups
         --timeout duration   Command timeout in seconds. (default 40s)

   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

   The response includes the unique ID of each backup. Note the ID of the backup that you want to restore, so you can use in it in the next step.

2. Restore a backup by running the following command with the appropriate flags:

   ```
   apicup subsys restore <analytics_subsystem>
   ```

   View the complete list of supported flags by running the `restore` command with the `-h` (help) flag, as shown in the following example:

   ```
   apicup subsys restore analytics -h
   Restore analytics subsystem

   Usage:
      restore SUBSYS [flags]

   Flags:
         --debug                   Enable debug logging
   ```

```
  -h, --help                    help for restore
      --id string               ID of the backup to restore
      --ignore-unavailable      If false, the restore will fail if an index is missing. If true, missing indices will be
skipped and the restore will continue.
      --indices strings         Indice(s) you'd like to restore (all/apievents/ui). It can be set to any combination of
indice(s) (eg. ui or ui,apievents) (default [all])
      --override                If set to true, then the restore operation will override existing indices.
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

- `--id` the backup ID provided must be enclosed in double quotes.
- `--indices` can be set to `all|apievents|ui|summary`, or the names of individual indices. `all` includes `apievents`, `ui`, and `summary` indices. Multiple values can be supplied with a comma-separated list. For example, to restore all UI data and all data from August of 2022:

  ```
  --indices "ui,apic-api-2022.08*"
  ```

- `--override` by default is set to `false` to prevent you from accidentally replacing existing indices. If `--override` is set to `false` and you specify indices that already exist in the database that you are restoring, then the restore fails. The failure prevents you from accidentally replacing or losing data that exists in the current database.

3. Optionally view a list of available restores, and the status of each, by running the following command:

   ```
   apicup subsys list-restores <analytics_subsystem>
   ```

   View the list of supported flags by running the `list-restores` command with the `-h` (help) flag, as shown in the following example:

   ```
   apicup subsys list-restores -h
   List restores of the subsystem

   Usage:
     apicup subsys list-restores SUBSYS [flags]

   Flags:
     -h, --help               help for list-restores
         --timeout duration   Command timeout in seconds. (default 40s)

   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

   Important:
   For deployment profiles with fewer than 3 storage nodes, the restore process is never marked as `Complete`. The storage cluster status is expected to be yellow and to have more than 0 unassigned shards. In this situation, the restore process is likely to be complete when there are 0 initializing shards.

Disaster recovery

Prepare for disaster events and recover API Connect on VMware if a disaster event occurs.

You must complete preparation steps prior to a disaster in order to recover API Connect on VMware.

The backups that you create for disaster recovery are for recovery of the API Connect subsystems in the same environment that the backups were taken, or in a different environment that has the same network configuration and API Connect form factor. If you want to move your API Connect deployment to a different environment and change the form factor or modify your API Connect endpoints, then see Migrating from v10 to v10 on a different form factor.
Note: For information about how to maintain a two data center deployment, including failover and recovery procedures, see Maintaining a two data center deployment.

- **Preparing for a disaster**
  Prepare your API Connect deployment for disaster recovery in a VMware environment.
- **Recovering from a disaster**
  Recover the API Connect subsystems on VMware.
- **Using VM snapshots for infrastructure backup and disaster recovery**
  You can use VM snapshots to backup and restore the infrastructure used by API Connect on VMware, and also for infrastructure disaster recovery.

Preparing for a disaster

Prepare your API Connect deployment for disaster recovery in a VMware environment.

Important: Successful disaster recovery of API Connect depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery.

- **Preparing the management subsystem for disaster recovery on VMware V10.0.6.0**
  Prepare a Management subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.
- **Preparing the Developer Portal subsystem for disaster recovery on VMware**
  Prepare a Developer Portal subsystem for disaster recovery by taking specific steps immediately after the installation of the subsystem.
- **Preparing the Analytics subsystem for disaster recovery on VMware**
  Prepare the API Connect Analytics subsystem for disaster recovery in a VMware environment.

Preparing the management subsystem for disaster recovery on VMware V10.0.6.0

Prepare a Management subsystem for disaster recovery by taking specific steps before and after the installation of the subsystem.

About this task

Note: This topic covers disaster recovery preparation for V10.0.6.0. For v10.0.7.0 see Backing up and restoring the Management subsystem.

- This task must be performed before any disaster event occurs, and also prior to the installation of a replacement Management subsystem during the recovery process. Best practice is to complete these steps immediately after initial configuration of a management subsystem during your original v10 deployment.
- Any `local` backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Make sure that you have a backup that can be used in case of a disaster event:
   - For backup configuration, see Configuring backup settings during initial installation of the management subsystem V10.0.6.0.
   - If scheduled backups were configured during backup configuration, you should have access to a backup on the specified cloud storage.
   - You can perform a manual backup at any time. See Backing up the management subsystem V10.0.6.0.
2. Ensure that you have a backup of your project directory.
   The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: First steps for deploying in a VMware environment. The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
3. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see Using VM snapshots for infrastructure backup and disaster recovery for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.
   Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

What to do next

You should now complete the preparation steps for the Developer Portal subsystem; see Preparing the Developer Portal subsystem for disaster recovery on VMware.

Preparing the Developer Portal subsystem for disaster recovery on VMware

Prepare a Developer Portal subsystem for disaster recovery by taking specific steps immediately after the installation of the subsystem.

About this task

- Perform this task before any disaster event occurs, and also prior to the installation of a replacement Developer Portal subsystem for the recovery process.
- Best practice is to complete these steps immediately after initial configuration of a Developer Portal subsystem during your original v10 deployment.
- Any `local` backups are presumed to have been lost in the disaster scenario and are non-recoverable in this procedure.
- Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. For information about preparing the Management subsystem, see Preparing the management subsystem for disaster recovery on VMware V10.0.6.0.
  If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Before the initial installation, configure the Developer Portal subsystem for database backups. See Configuring backup settings for the Developer Portal subsystem.
2. Ensure you create a backup:
   - You can perform a manual backup at any time. See Backing up the Developer Portal subsystem.
   - If you configure scheduled backups as part of Configuring backup settings for the Developer Portal subsystem, backups to the specified cloud storage take place at the configured time.
3. Ensure that you have a backup of your project directory.
   The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: First steps for deploying in a VMware environment. The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
4. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see Using VM snapshots for infrastructure backup and disaster recovery for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.
   Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

What to do next

To restore the Developer Portal on VMware after a disaster, see Recovering the Developer Portal subsystem on VMware.

Preparing the Analytics subsystem for disaster recovery on VMware

Prepare the API Connect Analytics subsystem for disaster recovery in a VMware environment.

About this task

Preparing for disaster recovery involves backing up analytics data and your project directory.

Procedure

1. Perform a backup of the Analytics database.
   Scheduling automatic backups ensures that a recovery includes recent updates. For more information on configuring backups, see Configuring backup settings for Analytics.

2. Ensure that you have a backup of your project directory.
   The original project directory that was created with the `apicup` command during the initial product installation is required for disaster recovery, see: First steps for deploying in a VMware environment. The project directory contains the yaml files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the subsystem databases on a deployment that uses a new or different project directory.
3. Ensure that you have a backup of the `analytics-extra-values.yaml` file if you created one: Analytics extra-values file. This file might be included in the project directory backup if you kept it there.
4. Optional: Take a Virtual Machine (VM) snapshot of all your VMs; see Using VM snapshots for infrastructure backup and disaster recovery for details. This action does require a brief outage while all of the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can offer a faster recovery when compared to redeploying OVAs and restoring from normal backups.
   Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

Recovering from a disaster

Recover the API Connect subsystems on VMware.

- **Recovering the Management subsystem on VMware V10.0.6.0**
  Recover the management subsystem from backups after a disaster event.
- **Recovering the Developer Portal subsystem on VMware**
  Recover the Developer Portal on VMware.
- **Recovering the Analytics subsystem on VMware**
  Recover the API Connect Analytics subsystem in a VMware environment.

Recovering the Management subsystem on VMware V10.0.6.0

Recover the management subsystem from backups after a disaster event.

Before you begin

Note: This topic covers disaster recovery for V10.0.6.0. For v10.0.7.0 see Recovering from a disaster.
To successfully recover the management subsystem, you must have previously completed the steps in Preparing the management subsystem for disaster recovery on VMware V10.0.6.0.
You must use the same project directory that you used for your original deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment.

In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.

Important: Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. You must complete preparation steps for both subsystems in order to achieve disaster recovery. If you have to perform a restore, you must complete the restoration of the Management Service first, and then immediately restore the Developer Portal. Therefore, the backups of the Management and Portal must be taken at the same time, to ensure that the Portal sites are consistent with Management database.

Procedure

1. Determine which backup to restore from.
   a. Obtain a list of the available backups for your backup type:
   - s3 backups
     - For IBM®'s Cloud Object Storage, you can check currently stored backups in the COS console. Select Buckets > Objects. See the Object Names displayed on the <*cos_name*>/backup/db panel. For example:

       ```
       20200605-105429F
       20200605-100008F
       20200605-11040F
       ```

     - For Amazon's AWS, you can check currently stored backups in the S3 console For example, under Amazon S3 > *cluster_name* > old cluster > backup > db:

       ```
       20200606-144315F
       20200606-145011F
       ```

   - SFTP backups
     View the SFTP backups available on your remote storage site:

```
-rw-r--r--    1 root     root     13092333 Aug 26 08:56 20200826-154646F.tgz
-rw-r--r--    1 root     root     18703758 Aug 26 09:10 20200826-160010F.tgz
-rw-r--r--    1 root     root     24318561 Aug 26 09:21 20200826-161301F.tgz
```

b. Select the backup ID of the backup you want to restore. For example, in the sample SFTP backup list, for the Aug 26 09:21 backup, the backup ID is **20200826-161301F**.
Each filename contains the date, time, and type of the backup stored. The format of the backup ID is **YYYYMMDD-HHMMSS<F|I>**. Full backups are denoted with a suffix **F** on the ID. Incremental backups are denoted with a suffix **I** on the ID. For incremental backups, ensure each incremental backup has its prior full backup also present in storage. You can check this by examining the ID **<prior-backup-id>_<backup-id>**.

Note:
During the disaster recovery process, the S3 configuration detail of the older management system is used, but the older management system must be in offline mode. The old subsystem must be offline because you cannot have two management systems simultaneously using the same s3 bucket name in the database backup configurations.

2. Make sure you know the management database cluster name. Use the following steps applicable to your backup type:

Backup type	How to obtain database cluster name
s3	a. Open your IBM Cloud® Object Storage or AWS S3 console and proceed to the bucket location where the old Management subsystem backups are located. b. Download **backup/db/<backup-id>/pg_data/postgresql.conf.gz**. Open **postgresql.conf** to view the database cluster name: • IBM Cloud Object Storage **cluster_name = 'm1-a6287572-postgres'** • **m1** - Management subsystem name • **a6287572** - site name • AWS S3 **cluster_name = 'm1-c91dc0b9-postgres'** • **m1** - Management subsystem name • **c91dc0b9** - site name
SFTP	a. Recover the Management database cluster name and siteName by examining the SFTP backup tar. Download or move the SFTP backup tar file and decompress (untar) it. b. Open **<management-subsystem-name>-<siteName>-postgres-backrest-shared-repo/backup/db/<backup-id>/pg_data/postgresql.conf.gz** which contains the management subsystem name and siteName. For example: **# Do not edit this file manually!** **# It will be overwritten by Patroni!** **include 'postgresql.base.conf'** **archive_command = 'source /opt/cpm/bin/pgbackrest/pgbackrest-set-env.sh && pgbackrest archive-push "%p"'** **archive_mode = 'True'** **archive_timeout = '60'** **autovacuum_vacuum_cost_limit = '1000'** **autovacuum_vacuum_scale_factor = '0.01'** **cluster_name = 'm1-f785a3e3-postgres'** In this example: • **cluster_name** has both the management subsystem name and siteName • **m1** - Management subsystem name • **f785a3e3** - site name

3. Use your prior existing project directory, or a restore of your project directory backup, to install the Management subsystem:
   a. Create your ISO files

   apicup subsys install *mgmt* --out *mgmt*plan-out

   The **--out** parameter and value are required.

   In this example, the ISO files are created in the myProject/*mgmt*plan-out directory.
   Note: If your original ISO files are still available and you did not upgrade from the original installation, you can reuse them. However, if you upgraded your original deployment, you must create new ISO files by using the version of **apicup** that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.6.x OVAs with ISO files that were created with **apicup** v10.0.5.3.
   b. Deploy the files into the replacement VMs. See Deploying the Management subsystem OVA file.
   c. Verify the deployment. See Verify installation of the Management subsystem.
   Important:
   For S3, the recovery remains in an intermediate state until the restore is complete, and Postgres wal files might cause serious disk issues. To avoid this possibility, continue immediately with the next step.

   Note that if you delay completion of the restore:

   - Health check might fail. In this case, you can still proceed to the next step and perform a restore.
   - Postgres wal files might cause problems by consuming all disk space. In this case, you must either:
     - Re-install the system, prepare again for disaster recovery, and perform the restore.
     - Or increase disk space so that the system returns to a stable state, and then proceed with the restore.
4. Once your Management subsystem is ready, confirm the backup ID noted in Step 1 is present on the sftp or s3 server.
5. After a few moments, confirm there is a **ManagementBackup** of type **record** and its backup ID matches with the backup ID noted in Step 1.
   You can list the management backups using:

   apicup subsys list-backups <subsystem_name>

   For example:

```
NAME                STATUS   ID                CLUSTER                          SUBSYSTEM    TYPE   CR TYPE   AGE
mgmt-backup-8hqqg   Ready    20200826-161301F  management-82b290a2-postgres     management   full   record    40s
```

6. Perform a Management Restore using the name of the backup that has the ID you want to restore.
   For example, for ID **20200826-161301F** the backup name is **mgmt-backup-8hqqg**.

   For instructions on how to restore, see Restoring the management subsystem V10.0.6.0.

   Once the Management Restore has completed and the database is running again, the data of the old Management subsystem will be successfully restored onto the new Management subsystem. Manual and scheduled backups should perform as normal once again.

What to do next

You should now complete the recovery steps for the Developer Portal subsystem on VMware, see Recovering the Developer Portal subsystem on VMware.

Recovering the Developer Portal subsystem on VMware

Recover the Developer Portal on VMware.

Before you begin

Successful disaster recovery depends on recovery of both the Management subsystem and the Developer Portal subsystem. If you must recover a deployment due to a disaster, you must first complete Recovering the Management subsystem on VMware V10.0.6.0, and then immediately recover the Developer Portal.

Recovery of the Portal subsystem is also dependent on prior completion of the preparation steps: Preparing the Developer Portal subsystem for disaster recovery on VMware

About this task

- In a disaster recovery scenario, an installation of a new Developer Portal subsystem is required.
- In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.
- You must use the same project directory that you used for your original Portal deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment. The **apiconnect-up-v10.yaml** file contains Portal configuration.

Use the following steps to recover the data from your previous Developer Portal subsystem.

Procedure

1. Use your prior existing project directory, or a restore of your project directory backup, to install the replacement Portal subsystem:
   a. Create your ISO files.

      ```
      apicup subsys install portal-subsystem-name --out portal-subsystem-nameplan-out
      ```

      The **--out** parameter and value are required.

      In this example, the ISO files are created in the myProject/*portal-subsystem-name*plan-out directory.
      Note: If your original ISO files are still available and you did not upgrade from the original installation, you can reuse them. However, if you upgraded your original deployment, you must create new ISO files by using the version of **apicup** that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.6.x OVAs with ISO files that were created with **apicup** v10.0.5.3.
   b. Deploy the files into the replacement VMs. See Deploying the Developer Portal subsystem OVA file.
   c. Verify the deployment. See Verifying deployment of the Developer Portal subsystem.
2. View the available backups and their status:

   ```
   apicup subsys list-backups [subsystem_name] [flags]
   ```

   Optional flags:

   ```
   ./apicup subsys list-backups -h
   List backups of the subsystem
   Usage:
     apicup subsys list-backups SUBSYS [flags]
   Flags:
     -h, --help               help for list-backups
         --timeout duration   Command timeout in seconds. (default 40s)
   Global Flags:
         --accept-license   Accept the license for API Connect
         --debug            Enable debug logging
   ```

3. Perform a Developer Portal restore.
   Run the following command, and attach any flags as appropriate.

   ```
   apicup subsys restore [SUBSYS_NAME] [flags]
   ```

   - To restore your Portal service, run the following command:

     ```
     apicup subsys restore ptl --timestamp [now|YYYYMMDD.HHMMSS]
     ```

     where **--timestamp** can be **now**, so the latest backup file that is available is used. Or you can specify a timestamp in the format of ***YYYYMMDD.HHMMSS*** to retrieve the nearest backup file to a specified time, searching backwards from the timestamp given. This command executes the portal restore process, which downloads the system backup and all of the site backups from the remote server, and installs them within the Portal stack. This process will then

restore the system configuration from the found backup, and restore all of the sites. Note that if a backed up site is already installed on the current stack, then the site is reinstalled by using the backup (the site is overwritten by the backup). If there are multiple sites to restore, then these sites are queued for restoring. You can track the restoration process within the Portal `www admin` logs.
Note: From IBM® API Connect Version 10.0.3.0, the timestamp format changed to ***`YYYYMMDD.HHMMSS`***. For Version 10.0.2.0 and earlier, the timestamp format is ***`YYYY-MM-DD HH:MM:SS`***.

- For a disaster recovery scenario, it is advised that you restore all sites and systems. By default the restore is a complete restore. You can optionally include `--type all` on the command line to specify a complete restore, but the flag is not required, as a complete restore is performed by default.
  Note that the complete restore uses the `priorityList` configuration setting that you specified in Configuring backup settings for the Developer Portal subsystem.
- For more information, see Restoring the Developer Portal subsystem

Recovering the Analytics subsystem on VMware

Recover the API Connect Analytics subsystem in a VMware environment.

About this task

Recovering the Analytics subsystem involves deploying a new installation of the Analytics subsystem and then restoring the Analytics database from a back-up copy.

You must use the same project directory that you used for your original deployment, or a restore of your project directory backup, to ensure that configuration and secret information is transferred to the replacement deployment.

In a clustered deployment, if any one VM is corrupted then all of the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.

Procedure

1. Use your prior existing project directory, or a restore of your project directory backup, to install the replacement Analytics subsystem:
   a. Create your ISO files:

      `apicup subsys install` ***`install-subsystem-name`*** `--out` ***`analyt`***`plan-out`

      The `--out` parameter and value are required.

      In this example, the ISO files are created in the myProject/*analyt*plan-out directory.
      Note: If your original ISO files are still available and you did not upgrade from the original installation, you can reuse them. However, if you upgraded your original deployment, you must create new ISO files by using the version of `apicup` that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.6.x OVAs with ISO files that were created with `apicup` v10.0.5.3.
   b. Deploy the files into the replacement VMs. See Deploying the Analytics subsystem OVA file.
   c. Verify the deployment. See Verifying deployment of the Analytics subsystem.
2. Restore the Analytics data from backup, as explained in Restoring the Analytics database.

Using VM snapshots for infrastructure backup and disaster recovery

You can use VM snapshots to backup and restore the infrastructure used by API Connect on VMware, and also for infrastructure disaster recovery.

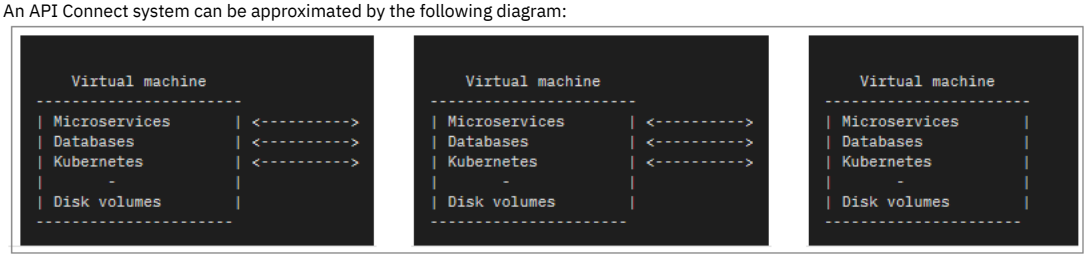
Backup and restore procedures for the databases used by IBM API Connect on VMware are described in Backing up and restoring on VMware. The procedures back up all of the state required by API Connect but do not include the underlying infrastructure software or other internal state.

In some scenarios, you may want to perform backups and disaster recovery at an infrastructure layer. You can do this by taking snapshots of the virtual machines or underlying storage volumes, providing that you follow the constraints described here.

Note that performing backups at the infrastructure layer differs from the standard approach in that not only is the database state backed up, but also the precise state of the software. This approach can be useful, for example, when testing upgrades of production systems.

VM snapshots should not be used as an alternative to taking normal backups. Ideal times to take VM snapshots are after initial install and configuration, and before and after upgrades. In the event of a disaster, reverting to the last snapshot taken and then restoring the standard daily backup can be faster than redeploying OVAs and restoring the backup.

- Requirements for taking a consistent backup of an API connect system at the infrastructure level
  - An API Connect system can be approximated by the following diagram:



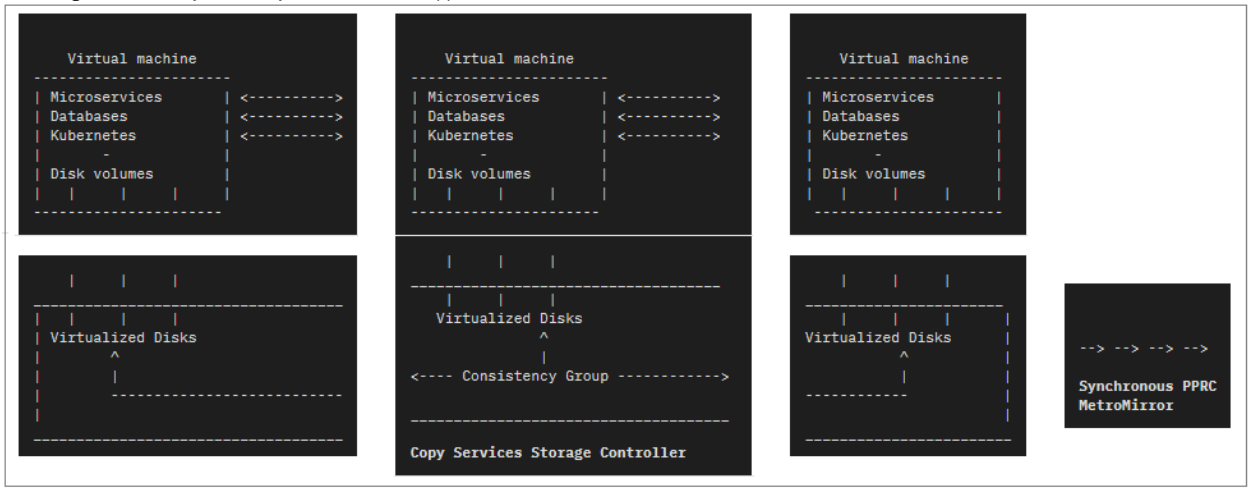
  - The system comprises multiple Virtual Machine images which are running Kubernetes, multiple databases, with the API Connect micro services running on top. There is transactional communication between the virtual machines which is occurring at all levels of this stack. This communication occurs all the time, even if the system is seemingly idle.

- Within the system there are multiple stateful or database layers on the software that maintain a consistency protocol. For example `etcd` upon which Kubernetes is based uses the Raft consensus algorithm. These algorithms depend for consistency on the basic assertion that time flows forwards on all systems together. If one or more of the virtual machine's state where to move backwards relative to the others by even the smallest quantum of time then consistency would be lost and the system might behave badly.

  Clearly, if Virtual Machine snapshots are taken across multiple virtual machines it is certain that these snapshots will not be taken at precisely the same time. The result will be a backup that contains a system state that contains data from snapshots taken at slightly different times. It is possible that such a backup system may be restored and may appear to work correctly but hidden deep within it there may be undetected inconsistencies that will cause strange unpredictable errors and inconsistencies later. These issues can be extremely difficult to diagnose.

  It is also possible that you can test this procedure successfully multiple times and see no apparent problem occurring. Irrespective of the apparent success of tests, corruption can be occurring in the system state. For this reason, taking VM snapshots or clones of a running API Connect system is not supported.

- How to take a consistent backup of an API connect system at the infrastructure level
  - The only way to take a consistent backup of an API connect system using VM snapshots and clones is to shut down ALL of the virtual machines comprising the system before taking the snapshot or clone.
  - Once the virtual machines are stopped at the VM level, time is effectively frozen. Then a snapshot can be taken on all the VMs. A clone can be made from the snapshot taken on each VM. This set of clones represent a valid, consistent state that the API Connect system was in when it was shut down.
  - Once all of the snapshots and clones have been taken the original API Connect system can be restarted.
- Restoring an API Connect System from VM clones
  - The Cloned VMs represent a valid state of the original API Connect system. They will restart in exactly the same way as the original system restarted.
  - If the objective is to stand up another instance of the cloned API connect system for testing or DR purposes, the following must be true:
    - The original system and the cloned system must be isolated from one another. The clone will be using the same hostnames and IP addresses as the original and so must be stood up in a separate virtual machine hosting environment with Network Address Translation between itself and any network to which the original system is stood up.
    - The cloned system must be stood up in an identical environment in terms of its hardware, software and network.
- Disaster Recovery approaches using Disk Replication:
  - A common DR approach used by enterprises is to use disk based replication. In this case, a storage controller such as IBM's San Volume Controller provides Copy Services. You can use Copy Services to create a completely consistent copy of a set of disks. IBM calls this feature *Synchronous Remote Copy* or *MetroMirror*. Other storage vendors have similar features.
  - If you use Remote Copy to synchronize disk images to a DR site then IBM will support API connect used with these systems so long as the copy at the DR site is completely consistent. Ensuring this consistency is the responsibility of the storage controller and its configuration. It is **critically important** to ensure that ALL of the disk volumes for all of the VMs or nodes in the entire API Connect system are in the same remote copy consistency group as shown in the following image. This ensures that DR scenarios do not create inconsistent volumes. If the disks are not in the same consistency group then the copy will not be managed consistently and the system will not be supported.



- Infrastructure based backup of API Connect using vSphere
  - The following procedure outlines steps to clone a VMware instance of API Connect v10, using the following topology:

    - 3 Management Nodes
    - 3 Portal Nodes
    - 3 Analytics Nodes
    - 3 Gateways
  - Each of these components has a load balancer with its own IP address. The load balancers all sit on a single VM. The cluster is installed and configured on 13 VMs in VMware. In this example, the system was populated with data and APIs. These should first be tested to ensure that all published APIs respond as expected. Once this is done, use vSphere UI to complete the following steps:

  1. Power off VMs
     To power off the VMs in sync, schedule a "shutdown guest OS" such that all the VMs shutdown at the same time. Although the shutdown does not occur at exactly the same instant, this does allow the shutdown for every machine to be as close in time as possible.

  2. Take snapshots
     Take snapshots of each VM after shutdown. Using the scheduling feature in the UI, schedule a snapshot of all the VMs at the same instant. If cloning should adversely affect the system in any way then the original VMs can be reverted back to these snapshots.

  3. Clone VMs
     Use the clone feature of the UI to create a clone of each VM. This VM should sit in the same resource pool as the original cluster. This is simply an exact copy of the disks of the original machine. The clones and originals are in no way coupled.

  4. Power on clones
     Schedule the power-on for each clone VM such that all the clone VMs power on simultaneously. Once this is done, test the cluster to ensure that the APIs respond as expected. For the example cluster topology shown earlier on this page, it can take a couple of hours after power-on for all the APIs to respond as expected.

Disabling and re-enabling the analytics subsystem on VMware

Disable the analytics subsystem by shutting down the VM that hosts it.

Before you begin

This operation requires the use of the Analytics CLI.

About this task

Disabling the analytics subsystem stops the collection and storage of data as well as making the subsystem unavailable.

When you disable analytics, there will be no data in either the API Manager Dashboards or third-party offloads.

Procedure

1. Back up your analytics data as explained in Backing up and restoring the Analytics database on VMware.
2. Shut down the analytics VMs.
   You can shut down the VMs using one of the following methods:
   - Use the VMware console
   - **`ssh`** into the images and run the **`shutdown`** command

Re-enabling analytics

3. Start your analytics VMs.

Maintaining a two data center deployment

How to maintain a two data center disaster recovery (DR) deployment on VMware, including information about normal operation, failure handling, and recovery.

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see Two data center warm-standby deployment strategy on VMware.

See the following topics for information about failure handling, backup, upgrade, and removing a 2DCDR deployment.

- **Backup and restore considerations for a two data center deployment**
  Points to consider when you configure backups and restoring a two data center disaster recovery (2DCDR) deployment on VMware.
- **How to failover a two data center deployment**
  If a failure occurs in your active data center, then failover to your warm-standby data center.
- **How to failover a two data center deployment V10.0.6.0**
  How to detect failures, and perform service failover, in a two data center disaster recovery deployment on VMware.
- **Removing a two data center deployment**
  You can remove your two data center disaster recovery deployment on VMware.

Related concepts

- Two data center warm-standby deployment strategy on VMware

Related tasks

- Installing a two data center deployment

Backup and restore considerations for a two data center deployment

Points to consider when you configure backups and restoring a two data center disaster recovery (2DCDR) deployment on VMware.

The 2DCDR backup requirements are different for the portal and management subsystems.

Management 2DCDR backup requirements

- Different remote backup locations must be specified for each data center.
- Both active and warm-standby management subsystems run the scheduled backup job.
- V10.0.7.0 and later: To restore a backup, 2DCDR must be disabled first.
- V10.0.6.0 only: Before you configure S3 backup settings you must first disable 2DCDR on both sites, and then re-enable 2DCDR after reconfiguring your backup settings.
  1. Disable 2DCDR for the management subsystems on both sites:

     ```
     apicup subsys set <management subsystem> multi-site-ha-enabled=false
     apicup subsys install <management subsystem>
     ```

  2. Configure your S3 backup settings.

3. Enable 2DCDR for the management subsystems on both sites:

```
apicup subsys set <management subsystem> multi-site-ha-enabled=true
apicup subsys install <management subsystem>
```

You do not need to do this for SFTP or local backups.

- V10.0.6.0 only: Restores must be run on the active data center.

Portal 2DCDR backup requirements

- The backup settings for both the active and warm-standby portal must be identical.
- Only the active data center portal takes backups. On failover, the backup job of the new warm-standby is disabled, and the backup job of the new active is enabled.
- Portal restores are only done on the active data center.

Configuring backups for 2DCDR deployments

Keeping the different portal and management 2DCDR backup requirements in mind, the actual steps for configuring backups for both subsystems are the same as for a single API Connect deployment: Backing up and restoring on VMware.

Related concepts

- Two data center warm-standby deployment strategy on VMware

Related tasks

- Installing a two data center deployment

How to failover a two data center deployment

If a failure occurs in your active data center, then failover to your warm-standby data center.

Note: The steps in this topic apply to API Connect V10.0.7.0 and later. If you are on V10.0.6.0, then refer to How to failover a two data center deployment V10.0.6.0.
Before you begin the failover process, first confirm that failover is the correct action to take.

Failover to the warm-standby is the appropriate action to take in the following circumstances:

- The failure is with the active data center itself, for example:
  - All VMs in the active data center are down.
  - Network connectivity to the active data is down.
- The API Connect deployment in the active data center is in a failed state. For example, the **health-check** command returns errors, and you are not able to resolve these errors.

  ```
  apicup subsys <subsystem name> health-check
  ```

The failover steps that you must follow depend on the condition of your active data center:

- **Failover steps when active site is responsive**
  Demote your active data center to warm-standby, and then promote your warm-standby data center to active.
- **Failover steps when active site is inaccessible**
  Disable network access to your failed active data center, and promote your warm-standby data center to active.

Failover steps when active site is responsive

Demote your active data center to warm-standby, and then promote your warm-standby data center to active.

About this task

Note: The steps in this topic apply to API Connect V10.0.7.0 and later. If you are on V10.0.6.0, then refer to How to failover a two data center deployment V10.0.6.0.
The steps in this topic cover how to complete a failover in the following scenarios:

- When the active management and portal subsystems are in a healthy state.
- The problem that the active management or portal subsystem has (that requires the failover) does not prevent the running of the **apicup** commands that are necessary to convert the active to warm-standby.

If the active site is not accessible, or responsive to **apicup** commands, then refer to Failover steps when active site is inaccessible.
For both management portal subsystems, the order in which failover is done is critical to prevent an active-active scenario that can cause a split-brain. The first step in failover is to demote the active data center to warm-standby, confirm that this demotion was successful, and then promote the original warm-standby to active.

In the steps provided in this topic, the active data center is called DC1, and the warm-standby data center is called DC2.

Procedure

- **Management subsystem failover**
  1. Set DC1 to be warm-standby.

a. Set the `multi-site-ha-mode` property to `passive` in DC1:

```
apicup subsys set <DC1 management> multi-site-ha-mode=passive
```

b. Apply the update to DC1:

```
apicup subsys install <DC1 management> --accept-dr-data-deletion
```

Note: When an active management subsystem is converted to warm-standby, all contents of its management database are deleted (to be replaced by the contents from the other data center when it becomes the active). The `--accept-dr-data-deletion` flag is acknowledgment that you accept this temporary loss of data.

c. Monitor the progress of the conversion to warm-standby:

```
apicup subsys health-check <DC1 management> -v
```

Important: If the management subsystem does not successfully convert to warm-standby, then do not proceed any further. Treat DC1 as an inaccessible data center, and refer to Failover steps when active site is inaccessible.

2. Set DC2 to be the active data center.
   a. Confirm that DC2 is ready for promotion to active:

```
apicup subsys health-check <DC2 management> -v
```

Output should show:

```
...
ReadyForPromotion
...
```

   b. Set the `multi-site-ha-mode` property to `active` in DC2:

```
apicup subsys set <DC2 management> multi-site-ha-mode=active
```

   c. Apply the update to DC2:

```
apicup subsys install <DC2 management> --skip-health-check
```

3. Monitor the failover process:

```
apicup subsys health-check <subsystem name>
```

When failover is complete the health-check command returns no output in either data center. Use the `-v` flag to see more information:

```
apicup subsys health-check <subsystem name> -v
```

4. Update your dynamic router to redirect all management subsystem traffic to DC2 instead of DC1.

- **Developer Portal service failover**

The following instructions show how to failover DC1 to DC2 for the Developer Portal service. If you have multiple portal services, then you must repeat these steps for each Developer Portal service that you want to failover.

1. Set DC1 to be warm-standby.
   a. Set the `multi-site-ha-mode` property to `passive` in DC1:

```
apicup subsys set <DC1 portal> multi-site-ha-mode=passive
```

   b. Apply the update to DC1:

```
apicup subsys install <DC1 portal> --skip-health-check
```

   c. Monitor the progression to warm-standby
      i. Login to one of the portal VMs in DC1 using SSH:

```
ssh apicadm@<portal hostname>
```

      ii. Switch to root user:

```
sudo -i
```

      iii. Run `kubectl describe ptl`, and look for `HA Mode` in the `Status` section of the output:

```
kubectl describe ptl

...
Status:
  ...
  HA Mode: progressing to passive
```

When `Status.HA Mode` shows `progressing to passive`, set portal in DC2 to be active, as described in step 2.

Warning: The `Spec` section of the output also has an `HA mode` property, take care to ensure that you are checking the `HA mode` in the `Status` section.

2. Set DC2 to be active.
   a. Set the `multi-site-ha-mode` property to `active` in DC2:

```
apicup subsys set <DC2 portal> multi-site-ha-mode=active
```

   b. Apply the update in DC2:

```
apicup subsys install <DC2 portal> --skip-health-check
```

3. Monitor the failover process:

```
apicup subsys health-check <subsystem name>
```

When failover is complete the health-check command returns no output in either data center. Use the `-v` flag to see more information:

```
apicup subsys health-check <subsystem name> -v
```

4. Update your dynamic router to redirect all traffic to DC2 instead of DC1.

Results

How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. However, here are some approximate timings:

For the management subsystem:

- `warm-standby` to `active` approximately 5 minutes
- `active` to `warm-standby` approximately 15 minutes

For the portal subsystem:

- `warm-standby` to `active` 15 - 40 minutes
- `active` to `warm-standby` approximately 10 minutes

Failover steps when active site is inaccessible

Disable network access to your failed active data center, and promote your warm-standby data center to active.

About this task

Note: The steps in this topic apply to API Connect V10.0.7.0 and later. If you are on V10.0.6.0, then refer to How to failover a two data center deployment V10.0.6.0.
Follow the steps in this procedure when your active site is inaccessible, or the API Connect subsystems in your active site are not responding to **apicup** operations.

Procedure

1. Update the network configuration in your failed data center to isolate the API Connect subsystems in this data center, so that they cannot communicate with each other, nor with the warm-standby data center. Network isolation is necessary to prevent a split-brain situation, which occurs if your active site recovers unexpectedly and starts communicating with your other subsystems.
2. Promote your warm-standby management subsystem to active:

   ```
   apicup subsys set <mgmt_subsystem> multi-site-ha-mode=active
   ```

   Apply the change with:

   ```
   apicup subsys install <mgmt_subsystem> --force-promotion --skip-health-check
   ```

   Monitor the progress of the promotion with:

   ```
   apicup subsys health-check <mgmt_subsystem>
   ```

   when the command returns no output, the promotion to active is complete.
   Use the `-v` flag to see more information:

   ```
   apicup subsys health-check <subsystem name> -v
   ```

3. Promote your warm-standby portal subsystem to active:

   ```
   apicup subsys set <portal_subsystem> multi-site-ha-mode=active
   ```

   Apply the change with:

   ```
   apicup subsys install <portal_subsystem>
   ```

   Monitor the progress of the promotion with:

   ```
   apicup subsys health-check <mgmt_subsystem>
   ```

   when the command returns no output, the promotion to active is complete.
   Use the `-v` flag to see more information:

   ```
   apicup subsys health-check <subsystem name> -v
   ```

4. Update your dynamic router to redirect all traffic to DC2 instead of DC1.

What to do next

If API Connect in your failed data center cannot be recovered, do not leave your remaining data center as a 2DCDR active with no functioning warm-standby. You have the following options:

- Convert your remaining data center to a stand-alone, and continue with one data center. See Removing a two data center deployment.
- Install a new warm-standby data center to replace the failed one. See Installing a two data center deployment.

If you are able to recover your failed data center, before you re-enable network access, set the management and portal subsystems to warm-standby:

1. Set the `multi-site-ha-mode` property to `passive` for the management subsystem in DC1:

```
apicup subsys set <DC1 management> multi-site-ha-mode=passive
```

2. Apply the update to DC1:

```
apicup subsys install <DC1 management> --accept-dr-data-deletion
```

Note: When an active management subsystem is converted to warm-standby, all contents of its management database are deleted (to be replaced by the contents from the other data center when it becomes the active). The `--accept-dr-data-deletion` flag is acknowledgment that you accept this temporary loss of data.

3. Monitor the progress of the conversion to warm-standby:

```
apicup subsys health-check <DC1 management> -v
```

4. Set the `multi-site-ha-mode` property to `passive` for the portal subsystem in DC1:

```
apicup subsys set <DC1 portal> multi-site-ha-mode=passive
```

5. Apply the update to DC1:

```
apicup subsys install <DC1 portal> --skip-health-check
```

6. Monitor the progress of the conversion to warm-standby:

```
apicup subsys health-check <DC1 portal> -v
```

7. When you have confirmed that both management and portal subsystems in DC1 are set to warm-standby, you can re-enable the network to the DC1 API Connect subsystems, and the two data centers should synchronize their API Connect data.
Monitor the health at both data centers with:

```
apicup subsys health-check <DC1 management> -v
```

How to failover a two data center deployment V10.0.6.0

How to detect failures, and perform service failover, in a two data center disaster recovery deployment on VMware.

Before you begin

Note: The steps in this task apply to API Connect V10.0.6.0. If you are on V10.0.7.0 or later, then refer to How to failover a two data center deployment.
Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see Two data center warm-standby deployment strategy on VMware.

About this task

Note:
The steps in this task apply to API Connect v10.0.6. If you are on v10.0.7 or later, then refer to How to failover a two data center deployment.

Failure handling comprises of two main concepts; failure detection, and failover of services.
Failure detection is about detecting that a failure has happened, and then alerting the relevant administrator for them to take action. Failure detection is different from system monitoring. Failure detection checks are run often and fast (for example, every 5 seconds). As such they are normally more limited in what they check. In API Connect, failure checks detect only whether the database and web server are online, and nothing more. Whereas monitoring checks typically are run less frequently, but are more likely to check for things like CPU, memory, and disk space usage. The results of these checks can then be recorded to perform historical trend analysis to spot memory leaks for example.

If a failure is detected, it's then important to make sure that the cluster or service is definitely offline. So, for example, if data center one (DC1) appears to have an outage, and the plan is to failover to data center two (DC2), then it's important to make sure that the pods in DC1 really are offline. If the DC1 pods are actually still running, then they might continue to send data to DC2, which would lead DC2 to think it wasn't the active deployment, and then that could lead to split-brain. A fast way to ensure that the cluster or service really is offline, is to drop the network link to those pods in your VMware infrastructure, and then delete them.

The failure detection endpoint must be location specific and must be exempt from any traffic routing, so that it's possible to explicitly check the health of each data center regardless of how traffic is being routed.

You can failover a specific service, for example `portal_service_1`, or all of the services that are running in a particular data center. The following sections describe how failover is achieved.

The operational states of an API Manager service:

Operational state	Description
`progressing to active`	Pods are progressing to the active state, but none are capable of serving traffic.
`active`	All of the pods are ready and in the correct disaster recovery state for the active data center.
`progressing to passive`	Pods are progressing to the warm-standby state, but none are capable of serving traffic.
`passive`	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.

The operational states of a Developer Portal service:

Operational state	Description
`progressing to active`	Pods are progressing to the active state, but none are capable of serving traffic.
`progressing to active (ready for traffic)`	At least one pod of each type is ready for traffic. The dynamic router can be linked to this service.
`active`	All of the pods are ready and in the correct disaster recovery state for the active data center.
`progressing to passive`	Pods are progressing to the warm-standby state, but none are capable of serving traffic.
`progressing to passive (ready for traffic)`	At least one pod of each type is ready for traffic.
`passive`	All of the pods are ready and in the correct disaster recovery state for the warm-standby data center.
`progressing to down`	Pods are moving from warm-standby state to down.
`down`	The multi-site HA mode is deleted from the warm-standby data center.

You can run the following command to check the operational state of a service:

```
kubectl describe ServiceName
```

Where `ServiceName` is the name of the API Manager or Developer Portal service.
Important:

- If the active data center is offline, do not change the `multi-site-ha-enabled` setting to `false` on the warm-standby data center, as this action deletes all the data from the databases.
  If you want to revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center, and then redeploy the warm-standby data center.

  If the active data center is offline, you must first change the warm-standby data center to be active, and then change the `multi-site-ha-enabled` setting to `false` on the now active data center.
- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center so you must revert your deployment to a single data center topology. To revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center. When the warm-standby site has been redeployed, the `multi-site-ha-enabled` setting can be reapplied to the active site.

Procedure

- **API Manager service failover**
  To initiate a failover from DC1 to DC2 you first must set DC1 to warm-standby, before making the warm-standby data center active. This action is needed to prevent split-brain, as there cannot be two active services or clusters at the same time. If the failover is being done because the current active data center has had a complete outage, then you should ensure that the network links to that data center are dropped in your VMware infrastructure, in order to avoid the potential for split-brain occurring. The offline data center must be set to warm-standby before the network links are restored. The following instructions show how to failover DC1 to DC2 for the API Manager service.

  1. Set DC1 (in this example, located in Dallas) to be warm-standby.
     Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Dallas data center, for example:

     ```
     apicup subsys set mgmt_dallas multi-site-ha-mode=passive
     ```

     where
     - `mgmt_dallas` is the name of the management service on DC1.

     Note: If DC1 is completely down, and you cannot set it to be warm-standby at this point, you must ensure that the network links to DC1 are removed before continuing to set DC2 to be active. You must then not restore the network links to DC1 until you can set DC1 to be warm-standby.
  2. Run `apicup` to update the settings on DC1, for example:

     ```
     apicup subsys install mgmt_dallas
     ```

  3. Log in to the virtual machine management subsystem on DC1 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

     ```
     kubectl describe ServiceName
     ```

     The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `passive`. For example:

     ```
     Status:
       ...
       Ha mode:                              passive
       ...
     ```

  4. Set DC2 (in this example, located in Raleigh) to be active.
     Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Raleigh data center, for example:

     ```
     apicup subsys set mgmt_raleigh multi-site-ha-mode=active
     ```

     where
     - `mgmt_raleigh` is the name of the management service on DC2.
  5. Run `apicup` to update the settings on the services on DC2.
     For example:

     ```
     apicup subsys install mgmt_raleigh
     ```

  6. Update your dynamic router to redirect all traffic to DC2 instead of DC1. Note that until the DC2 site becomes `active`, the UIs might not be operational.
  7. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

     ```
     kubectl describe ServiceName
     ```

     The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `active`. For example:

     ```
     Status:
       ...
       Ha mode:                              active
       ...
     ```

- **Developer Portal service failover**
  To initiate a failover for the Developer Portal from DC1 to DC2 you first must ensure that the existing DC1 is warm-standby, before making the warm-standby data center active, in order to prevent split-brain. The following instructions show how to failover DC1 to DC2 for the Developer Portal service. You must repeat these steps for each Developer Portal service that you want to failover.

  Note:
  - Switching a Developer Portal service to warm-standby mode instantly restarts all of its `portal-db` and `portal-www` pods at the same time. Therefore, if you don't have an active Developer Portal service that is online, there will be an outage of the Developer Portal web sites until the warm-standby data center is ready to accept traffic.

- If you want to monitor the failover process, you can run the following command to check the status of the Portal pods in the ***PortalService*** CR file:

  ```
  kubectl describe ServiceName
  ```

  For more details about the status information in a **PortalCluster** CR file, see the Example section.
- Do not edit the ***PortalService*** CR file during the failover process.

1. Set DC1 (in this example, located in Dallas) to be warm-standby.
   Run **apicup** to change the **multi-site-ha-mode** property to **passive** on the Dallas data center, for example:

   ```
   apicup subsys set port_dallas multi-site-ha-mode=passive
   ```

   where
   - ***port_dallas*** is the name of the Portal service on DC1.

   Note: If DC1 is completely down, and you cannot set it to be warm-standby at this point, you must ensure that the network links to DC1 are removed before continuing to set DC2 to be active. You must then not restore the network links to DC1 until you can set DC1 to be warm-standby.
2. Run **apicup** to update the settings on DC1, for example:

   ```
   apicup subsys install port_dallas --skip-health-check
   ```

   Important: You must not wait for the active DC **Status** to become **warm-standby**, before failing over the warm-standby DC to become **active**. You should change the DC1 **multi-site-ha-mode** to **passive**, and then run the **kubectl describe *ServiceName*** command, and as soon as the **haMode** is set to **progressing to passive**, you must change DC2 to be **active**.
3. Set DC2 (in this example, located in Raleigh) to be active.
   Run **apicup** to change the **multi-site-ha-mode** property to **active** on the Raleigh data center, for example:

   ```
   apicup subsys set port_raleigh multi-site-ha-mode=active
   ```

   where
   - ***port_raleigh*** is the name of the Portal service on DC2.
4. Run **apicup** to update the settings on the services on DC2.
   For example:

   ```
   apicup subsys install port_raleigh --skip-health-check
   ```

5. Update your dynamic router to redirect all traffic to DC2 instead of DC1.
6. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the Developer Portal service is ready for traffic:

   ```
   kubectl describe ServiceName
   ```

   The service is ready for traffic when the **haMode** part of the **Status** object is set to **active**. For example:

   ```
   Status:
     ...
     haMode:                              active
     ...
   ```

- **Entire data center failover**
  To failover an entire datacenter, follow the previous steps to failover the API Manager service, followed by the steps to failover each Developer Portal service in that data center.

  How long it takes to complete the failover varies, and depends on hardware speed, network latency, and the size of the databases. However, here are some approximate timings:

  For API Manager:
  - **warm-standby** to **active** approximately 5 minutes
  - **active** to **warm-standby** approximately 15 minutes

  For Developer Portal:
  - **warm-standby** to **active** 15 - 40 minutes
  - **active** to **warm-standby** approximately 10 minutes

Example

Example of the status section of a **PortalCluster** CR file for an active data center that you can view by running the **kubectl describe *PortalService*** command:

```
status:
  conditions:
  - lastTransitionTime: "2020-04-27T08:00:25Z"
    message: 4/6
    status: "False"
    type: Ready
  customImages: true
  dbCASecret: portal-db-ca
  encryptionSecret: ptl-encryption-key
  endpoints:
    portal-director: https://api.portal.ha-demo.cloud/
    portal-web: https://portal.ha-demo.cloud/
    replication: https://ptl-replication.cvs.apic-2ha-dallas-144307-xxxx14539d6e75f74e950db7077951d4-0000.us-
south.containers.appdomain.cloud/
  localMultiSiteHAPeerConfigHash: UM9VtuWDdp/328n0oCl/GQGFzy34o1V9sH2OyXxKMHw=
  haMode: progressing to active (ready for traffic)
  microServiceSecurity: custom
  multiSiteHA:
    dbPodScale: 3
    remoteSiteDeploymentName: portal
    remoteSiteName: frankfurt
    replicationPeer: https://ptl-replication.cvs.apic-2ha-frankfur-683198-xxxx14539d6e75f74e950db7077951d4-0000.eu-
```

```
de.containers.appdomain.cloud/
    wwwPodScale: 3
  phase: 0
  serviceCASecret: portal-ca
  serviceClientSecret: portal-client
  serviceServerSecret: portal-server
  services:
    db: portal-dallas-db
    db-remote: portal-frankfurt-db
    nginx: portal-nginx
    tunnel: portal-tunnel
    www: portal-dallas-www
    www-remote: portal-frankfurt-www
  versions:
    available:
      channels:
      - name: "10"
      - name: "10.0"
      - name: 10.0.0
      - name: 10.0.0.0
      versions:
      - name: 10.0.0.0-1038
    reconciled: 10.0.0.0-1038
```

Meaning of the **haMode** for the active data center:

- **active** - this is the active data center, or the only data center, and all pods are ready and accepting traffic.
- **progressing to active** - this is the active data center, or the only data center, and there are no pods ready yet.
- **progressing to active (ready for traffic)** - this is the active data center, or the only data center, and there is at least one pod of each type, www, db, and nginx, ready, so the data center is accepting traffic.

Example of the status section of a **PortalCluster** CR file for a warm-standby data center that you can view by running the **kubectl describe *PortalService*** command:

```
status:
  conditions:
  - lastTransitionTime: "2020-04-21T10:54:54Z"
    message: 6/6
    status: "True"
    type: Ready
  customImages: true
  dbCASecret: portal-db-ca
  encryptionSecret: ptl-encryption-key
  endpoints:
    portal-director: https://api.portal.ha-demo.cloud/
    portal-web: https://portal.ha-demo.cloud/
    replication: https://ptl-replication.cvs.apic-2ha-frankfur-683198-xxxx14539d6e75f74e950db7077951d4-0000.eu-
de.containers.appdomain.cloud/
  localMultiSiteHAPeerConfigHash: VguODE74TkiS3LCc5ytQiaF8100PXMHUrVBtb+PbKOg=
  haMode: progressing to passive (ready for traffic)
  microServiceSecurity: custom
  multiSiteHA:
    dbPodScale: 3
    remoteSiteDeploymentName: portal
    remoteSiteName: dallas
    replicationPeer: https://ptl-replication.cvs.apic-2ha-dallas-144307-xxxx14539d6e75f74e950db7077951d4-0000.us-
south.containers.appdomain.cloud/
    wwwPodScale: 3
  phase: 0
  serviceCASecret: portal-ca
  serviceClientSecret: portal-client
  serviceServerSecret: portal-server
  services:
    db: portal-frankfurt-db
    db-remote: portal-dallas-db
    nginx: portal-nginx
    tunnel: portal-tunnel
    www: portal-frankfurt-www
    www-remote: portal-dallas-www
  versions:
    available:
      channels:
      - name: "10"
      - name: "10.0"
      - name: 10.0.0
      - name: 10.0.0.0
      versions:
      - name: 10.0.0.0-1038
    reconciled: 10.0.0.0-1038
```

Meaning of the **haMode** for the warm-standby data center:

- **passive** - this is the warm-standby data center, and all pods are ready and accepting traffic.
- **progressing to passive** - this is the warm-standby data center, and there are no pods ready yet.
- **progressing to passive (ready for traffic)** - this is the warm-standby data center, and there is at least one pod of each type, www, db, and nginx, ready, so the data center is accepting traffic.

What to do next

As soon as a failure on a data center has been resolved, the failed data center should be brought back online and re-linked to the currently active data center in order to maintain the highest availability; see Recovering from a failover of a two data center deployment V10.0.6.0 for more information.

- **Recovering from a failover of a two data center deployment V10.0.6.0**
  How to recover a two data center disaster recovery deployment on VMware.

Related concepts

- Two data center warm-standby deployment strategy on VMware

Related tasks

- Recovering from a failover of a two data center deployment V10.0.6.0
- Installing a two data center deployment

Recovering from a failover of a two data center deployment V10.0.6.0

How to recover a two data center disaster recovery deployment on VMware.

Before you begin

Note: The steps in this task apply to API Connect V10.0.6.0. If you are on V10.0.7.0 or later, then refer to How to failover a two data center deployment.
Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see Two data center warm-standby deployment strategy on VMware.

About this task

After a failure has been resolved, the failed data center can be brought back online and re-linked to the currently active data center. It's important to do this as soon as possible, in order to reinstate disaster recovery. Otherwise, if another failure occurred before the failed data center is brought back online, there could be a complete outage.

The failed data center can be brought back online as the new active primary data center, or it can be brought back as a warm-standby secondary data center and the currently active data center kept as the primary one. This decision will be based on your own company policies about recovering from a failure.

The amount of data that is sent when recovering from a failover, will depend on how long the outage lasted, and how much activity there was during that period.

Important:

- If the active data center is offline, do not change the `multi-site-ha-enabled` setting to `false` on the warm-standby data center, as this action deletes all the data from the databases.
  If you want to revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center, and then redeploy the warm-standby data center.

  If the active data center is offline, you must first change the warm-standby data center to be active, and then change the `multi-site-ha-enabled` setting to `false` on the now active data center.
- If the warm-standby data center is offline for more than 24 hours, there can be issues with the disk space on the active data center so you must revert your deployment to a single data center topology. To revert to a single data center topology, you must change the `multi-site-ha-enabled` setting to `false` on the active data center. When the warm-standby site has been redeployed, the `multi-site-ha-enabled` setting can be reapplied to the active site.

Procedure

- **Recovering from an API Manager failover**
  The following instructions show how to bring data center one (DC1) back as the primary (`active`) data center, and assume that DC1 is now online and in the warm-standby state. If you prefer, you can keep DC1 as the secondary data center, and leave data center two (DC2) as the primary (`active`) data center.

  1. Set DC2 (in this example, located in Raleigh) to be warm-standby.
     Run `apicup` to change the `multi-site-ha-mode` property to `passive` on the Raleigh data center, for example:

     ```
     apicup subsys set mgmt_raleigh multi-site-ha-mode=passive
     ```

     where
     - *`mgmt_raleigh`* is the name of the management service on DC2.
  2. Run `apicup` to update the settings on DC2, for example:

     ```
     apicup subsys install mgmt_raleigh
     ```

  3. Log in to the virtual machine management subsystem on DC2 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

     ```
     kubectl describe ServiceName
     ```

     The service is ready for traffic when the `Ha mode` part of the `Status` object is set to `passive`. For example:

     ```
     Status:
       ...
       Ha mode:                          passive
       ...
     ```

     You must also ensure that the database in DC2 has replicated over completely before proceeding.
  4. Set DC1 (in this example, located in Dallas) to be active.
     Run `apicup` to change the `multi-site-ha-mode` property to `active` on the Dallas data center, for example:

```
apicup subsys set mgmt_dallas multi-site-ha-mode=active
```

where
- *mgmt_dallas* is the name of the management service on DC1.

5. Run **apicup** to update the settings on DC1.
   For example:

```
apicup subsys install mgmt_dallas
```

6. Update the dynamic router to redirect all traffic to DC1 instead of DC2. Note that until the DC1 site becomes **active**, the UIs might not be operational.
7. Log in to the virtual machine management subsystem on DC1 by using an SSH tool, and run the following command to check that the API Manager service is ready for traffic:

```
kubectl describe ServiceName
```

   The service is ready for traffic when the **Ha mode** part of the **Status** object is set to **active**. For example:

```
Status:
  ...
  Ha mode:                          active
  ...
```

- **Recovering from a Developer Portal failover**
  When recovering from a Developer Portal failover, how much data is sent from the now warm-standby data center (in this example, DC2) back to the newly recovered active data center (in this example, DC1) varies depending on how much time has passed and how many changes have been made in the meantime.

  If possible, the Portal sends an Incremental State Transfer (IST), which is a replay of the transactions that have happened since the failed data center, DC1, became unreachable. The IST can grow to be up to 3 GB in size. If the IST grows larger than 3 GB, then the Portal sends a State Snapshot Transfer (SST) instead, which is basically a full copy of the whole database. How long this process takes depends on how many Portal sites there are, how much content in each of those sites, how active they are (as in number of transactions), and how fast the network speed is between the data centers. However, it is likely to take in the range 5 - 60 minutes.

  You can check whether the Developer Portal servers are in sync by running the **status** command in the **portal-www** pod admin container. If all servers show as **primary**, then they are in sync and traffic can be directed wherever needed. For information about running the status command, see Verifying deployment of the Developer Portal subsystem.

  Note:
  - If you want to monitor the recovery process, you can log in to the virtual machine Portal subsystem by using an SSH tool, and run the following command to check the status of the Portal pods in the ***PortalService*** CR file:

```
kubectl describe ServiceName
```

    For more details about the status information in a **PortalCluster** CR file, see the Example section.
  - Do not edit the ***PortalService*** CR file during the recovery process.

  The following instructions show how to bring DC1 back as the active primary data center, and switch DC2 to be the warm-standby secondary data center. Again, you can keep DC2 as the active primary data center if you prefer, and leave DC1 to be warm-standby.

  1. Set DC2 (in this example, located in Raleigh) to be warm-standby.
     Run **apicup** to change the **multi-site-ha-mode** property to **passive** on the Raleigh data center, for example:

```
apicup subsys set port_raleigh multi-site-ha-mode=passive
```

     where
     - *port_raleigh* is the name of the Portal service on DC2.

  2. Run **apicup** to update the settings on the services on DC2.
     For example:

```
apicup subsys install port_raleigh --skip-health-check
```

     Important: You must not wait for the active DC **Status** to become **warm-standby**, before changing the warm-standby DC to become **active**. You should change the DC2 **multi-site-ha-mode** to **passive**, and then run the **kubectl describe** *ServiceName* command, and as soon as the **haMode** is set to **progressing to passive**, you must change DC1 to be **active**.
  3. Set DC1 (in this example, located in Dallas) to be active.
     Run **apicup** to change the **multi-site-ha-mode** property to **active** on the Dallas data center, for example:

```
apicup subsys set port_dallas multi-site-ha-mode=active
```

     where
     - *port_dallas* is the name of the Portal service on DC1.

  4. Run **apicup** to update the settings on the services on DC1.
     For example:

```
apicup subsys install port_dallas --skip-health-check
```

  5. Update the dynamic router to redirect all traffic to DC1 instead of DC2.
  6. Log in to the virtual machine Portal subsystem on DC1 by using an SSH tool, and run the following command to check that the Developer Portal service is ready for traffic:

```
kubectl describe ServiceName
```

     The service is ready for traffic when the **Ha mode** part of the **Status** object is set to **active**. For example:

```
Status:
  ...
  Ha mode:                          active
  ...
```

Related concepts

- Two data center warm-standby deployment strategy on VMware

Related tasks

- How to failover a two data center deployment V10.0.6.0
- Installing a two data center deployment

Removing a two data center deployment

You can remove your two data center disaster recovery deployment on VMware.

Before you begin

Ensure that you understand the concepts of a 2DCDR deployment in API Connect. For more information, see Two data center warm-standby deployment strategy on VMware.

About this task

If you decide to remove your two data center disaster recovery (DR) deployment, you go from two data centers to one. The active data center remains as the only working system. The warm-standby data center has its data deleted.

Warning: If the active data center is offline, do not disable 2DCDR in the warm-standby data center, as this action deletes all the data from the management and portal databases.
As part of the installation, you configured the `multi-site-ha` settings on each data center, as described in the following table:

Table 1. The `multi-site-ha` settings

Setting	Description
`multi-site-ha-enabled`	Indicates whether two data center DR deployment is enabled. Set to `true` to enable two data center DR. Set to `false` to disable two data center DR. The default value is `false`.
`multi-site-ha-mode`	The state of the data center in the context of the two data center deployment. Valid values are: • `active` - indicates the primary data center • `passive` - indicates the warm-standby data center
`management-replication`	Required only for the management subsystem. The external ingress name for the management subsystem in the current data center in the two data center deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
`portal-replication`	Required only for the portal subsystem. The external ingress name for the portal subsystem in the current data center in the two data center deployment. The name is a unique fully qualified hostname that the other data center uses to communicate with the current data center.
`replication-peer-fqdn`	The ingress hostname for the other data center in the two data center deployment. This information is required so that the two data centers can communicate with each other.

If you want to revert to a single data center topology, or temporarily remove the warm-standby data center from DR, you must disable the `multi-site-ha` configuration on the active data center, and then remove the warm-standby data center. If the active data center is offline, you must first change the warm-standby data center to be active, and then disable the `multi-site-ha` configuration on the now active data center.

Procedure

- Disable the `multi-site-ha` configuration on the active data center:
  1. Disable the `multi-site-ha` configuration on the management subsystem of the active data center:

     ```
     apicup subsys set mgmt multi-site-ha-enabled=false
     apicup subsys install mgmt
     ```

  2. Disable the `multi-site-ha` configuration on the portal subsystem of the active data center:

     ```
     apicup subsys set ptl multi-site-ha-enabled=false
     apicup subsys install ptl
     ```

- Disable the `multi-site-ha` configuration on the warm-standby data center:
  Note:
  Disabling 2DCDR on the warm-standby results in deletion of all data on the warm-standby.

  If you do not plan to keep the warm-standby installation, you can uninstall API Connect on the warm-standby, you do not need to convert it to a stand-alone before you uninstall.
  1. Disable the `multi-site-ha` configuration on the management subsystem of the warm-standby data center:

     ```
     apicup subsys set mgmt multi-site-ha-enabled=false
     apicup subsys install mgmt --accept-dr-data-deletion
     ```

  2. Disable the `multi-site-ha` configuration on the portal subsystem of the warm-standby data center:

     ```
     apicup subsys set ptl multi-site-ha-enabled=false
     apicup subsys install ptl
     ```

Related concepts

- Maintaining a two data center deployment
- Backup and restore considerations for a two data center deployment

Related tasks

- How to failover a two data center deployment V10.0.6.0
- Recovering from a failover of a two data center deployment V10.0.6.0

Monitoring and health checks

Monitor your deployment and complete health checks on subsystems.

See:

- **Checking cluster health on VMware**
  You can use `apicup` to check the health of the API Connect clusters in your VMware deployment.
- **Monitoring Postgres disk usage on VMware**
  Monitor disk usage by the Postgres database in the management subsystem, when deployed on VMware.
- **Obtaining simple health check data of Developer Portal sites by using a REST API call**
  Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.
- **Monitoring the network with SNMP**
  Presents a table of OID trees that you can poll using SNMP Get.

Checking cluster health on VMware

You can use **apicup** to check the health of the API Connect clusters in your VMware deployment.

The **health-check** command checks a number of criteria to determine the health of their cluster. When all criteria are successfully met, the command displays no output, and exits with a status of **0**. When one or more criteria are not met, the command stops processing and displays a message with the failure and exits with a status of **1**.

The command takes no arguments. The only option is the **--verbose** flag. If **--verbose** is set the command prints all checks that were performed.

The **health-check** is run against the namespace that is specified in apiconnect-up.yaml.

The health check is run on the specified subsystem.

The command verifies that for the specified subsystem:

- The **apicup** version matches the API Connect release version.
- All cluster members are running the same API Connect release version.
- All cluster members have a deployment status of **Done**.
- Docker is running.
- **kubelet** is running
- All Kubernetes-defined nodes are running.
- The Kubernetes Control Plane pods are running:
  - **etcd/kube-apiserver**
  - **kube-system.kube-controller**
  - **kube-scheduler**
  - **kube-apiserver-proxy**
- The add-on Kubernetes Deployments are fulfilled:
  - **coredns**
  - **metrics-server**
- The add-on Kubernetes DaemonSets are fulfilled:
  - **calico-node**
  - **ingress-nginx-ingress-controller**
  - **kube-proxy**

Syntax:

```
Usage:
  apicup subsys health-check <SUBSYS>  [flags]

Flags:
  -h, --help                help for health-check
      --kubeconfig string   (optional) absolute path to the kubeconfig file (default "/Users/<username>/.kube/config")
  -v, --verbose             Verbose output

Global Flags:
      --accept-license   Accept the license for API Connect
      --debug            Enable debug logging
```

Example usage, for a subsystem named **mgmt**:

```
../apicup subsys health-check mgmt
```

Note:

- The command flag **--kubeconfig string** does not apply to deployments on VMware.

- In a multi-node OVA cluster, the **apicup subsys health-check** command does not return status information if a majority of the nodes are down. To obtain status information, start more nodes. Status is returned only when a quorum is achieved.

Monitoring Postgres disk usage on VMware

Monitor disk usage by the Postgres database in the management subsystem, when deployed on VMware.

Postgres Database is the core database used in Management Subsystem. It is important to monitor the Postgres disk usage.

The APIConnect operator tracks the current disk usage of the Postgres components, and regularly updates **apic health-check** command.

APIConnect operator regularly updates the **ManagementCluster** instance status which is reflected in the **apic health-check** command.

apic health-check command reports WARNING state when disk utilization reaches 60%.

apic health-check command reports ERROR state when disk utilization reaches 70% and Postgres is brought down.

Management Cluster disk stats

The **ManagementCluster** instance has **.status.postgresDataStats** where the operator prints the current disk usage of Postgres components. Use **kubectl** to view the status.

To use **kubectl**, first ssh into the appliance as **root**.

```
kubectl get mgmt -o json | jq .status.postgresDataStats
[
  {
    "instanceName": "management-site1-db-1",
    "podName": "",
    "pvcCapacity": 51200,
    "pvcName": "",
    "pvcType": "WAL",
    "pvcUsed": 400,
    "pvcUsedPercentage": 8
  },
  {
    "instanceName": "management-site1-db-1",
    "podName": "",
    "pvcCapacity": 184320,
    "pvcName": "",
    "pvcType": "PostgreSQL",
    "pvcUsed": 95,
    "pvcUsedPercentage": 8
  }
]
```

Warning condition is populated at 60% usage

When one of the Postgres components occupy 60% usage, the operator marks the overall status of the **ManagementCluster** to **Warning** with an appropriate warning message, which is included in the output from **apic health-check** command.

The warning condition explains why the warning condition was set.

To use **apic**, first ssh into the appliance as **root**.

```
apic health-check
```

```
FATA[0007] Cluster not in good health:
ManagementCluster is not Ready or Complete | State: 17/17 Phase:
  Warning Message: Current WAL disk usage of nihar-stac-3955b42d-site1-postgres is 61 percent.
  DATABASE SHUTDOWN starts at 70 percent utilization. Please contact IBM Support immediately
```

If you encounter the warning, contact IBM support to fix the root cause.

Error condition is populated at 70% usage

An error condition is populated when the operator brings down Postgres because it is using 70% of available disk space. Postgres is brought down to avoid problems, such as disk corruption, that can occur if the disk becomes completely filled.

To use **apic**, first ssh into the appliance as **root**.

```
apic health-check
```

```
FATA[0011] Cluster not in good health:
ManagementCluster is not Ready or Complete | State: 17/17 Phase:
  Error Message: Current WAL disk usage of nihar-stac-3955b42d-site1-postgres is more than 70 percent,
  initiating DATABASE SHUTDOWN
```

If you encounter the error, contact IBM support to fix the root cause.

Important:
On VMware, because the entire disk is allocated to the worker node, in some cases before the operator reacts to 70% usage, , Kubernetes itself can face disk pressure and starts evicting pods. In these cases, increase the disk allocated to worker node in order to stabilize the worker node. See Adding disk space to a VMware appliance.

Obtaining simple health check data of Developer Portal sites by using a REST API call

Call a simple health check API from your external load balancer to dynamically determine whether a specific Developer Portal site in a cluster is working. This API call can be used by a load balancer to help determine where to route traffic.

About this task

You can use the site health REST API to determine whether a particular Developer Portal site is running. The site health API returns the current system time of the site if both the database and web server are running. This API is fast and puts no load on the system, so it is ideal for use with load balancers to help them determine where to route traffic.

Procedure

To call the site health REST API, append **`/health`** to the end of your Developer Portal site URL in your web browser, as follows:

***`site_url`*`/health`**

Where ***`site_url`*** is the URL of the Developer Portal site that you want to check.
If both the database and web server of the site are running, the web browser returns the current system time. For example:

```
1511367695
```

If either the database or the web server of the site is not running, the web browser returns an error that the site can't be reached.
If you do get an error from the **`/health`** endpoint, refer to Gathering logs for a VMware environment and obtain the **`portal-www web`** logs, so that you can see more information on the error.

Monitoring the network with SNMP

Presents a table of OID trees that you can poll using SNMP Get.

Simple Network Management Protocol (SNMP) collects information from network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP). SNMP employs addresses for network devices using a hierarchical numbering system called Object Identifiers (OID). The following table provides OID addresses for polling in SNMP to monitor API Connect servers.

Note:

- Only SNMP v2 and SNMP v3 are supported. However, it is not recommended to enable SNMP v2. Instead, configure SNMPv3 as shown in Configuring SNMPv3, and then enable it as shown in Enabling SNMP.
- If the server has two or more network interfaces, the SNMP request is sent only to the first network interface.
- SNMP is disabled by default for all hosts.
- This information applies to API Connect deployments in VMware environments. It does not apply to Kubernetes deployments.

Each entry in the table represents many individual items. Use snmpwalk or another SNMP polling utility to see the complete list. You can poll the following OID trees:

Table 1. Default SNMPv2 configuration

OID	SNMP Name	Notes
.1.3.6.1.2.1.1	SNMPv2-MIB::system	
.1.3.6.1.2.1.2	IF-MIB::interfaces	
.1.3.6.1.2.1.4	IP-MIB::ip	
.1.3.6.1.2.1.5	IP-MIB::icmp.	
.1.3.6.1.2.1.6	TCP-MIB::tcp	
.1.3.6.1.2.1.7	UDP-MIB::udp	
.1.3.6.1.2.1.11	SNMPv2-MIB::snmp	
.1.3.6.1.2.1.25.1	HOST-RESOURCES-MIB::hrSystem	Excluding .1.3.6.1.2.1.25.1.3 HOST-RESOURCES-MIB::hrSystemInitialLoadDevice Excluding .1.3.6.1.2.1.25.1.4 HOST-RESOURCES-MIB::hrSystemInitialLoadParameters
.1.3.6.1.2.1.25.2	HOST-RESOURCES-MIB::hrStorage	
.1.3.6.1.2.1.25.3	HOST-RESOURCES-MIB::hrDevice	
.1.3.6.1.4.1.2021.4	UCD-SNMP-MIB::memory	
.1.3.6.1.4.1.2021.10	UCD-SNMP-MIB::laTable	CPU Load Average
.1.3.6.1.4.1.2021.11	UCD-SNMP-MIB::systemStats	

Configuring SNMPv3

From an OVA shell prompt as root edit the file **`/etc/snmp/snmpd.conf`** so that its content is:

```
createUser readOnlyUser SHA authpassphrase AES privpassphrase
rouser readOnlyUser
agentAddress udp:161,udp6:[::1]:161
```

- Replace ***`authpassphrase`*** with the passphrase to be used for authentication,
- Replace ***`privpassphrase`*** with the passphrase to be used for privacy (encryption) of the communications.

Enabling SNMP

It is not recommended to start SNMP without first configuring it for SNMPv3. See Configuring SNMPv3.

Enable SNMP on each individual OVA node with the following commands from an OVA shell prompt as root:

```
systemctl start snmpd
systemctl enable snmpd
```

Sample output:

```
root@mgmt:~# systemctl start snmpd
root@mgmt:~# systemctl enable snmpd
snmpd.service is not a native service, redirecting to systemd-sysv-install
Executing /lib/systemd/systemd-sysv-install enable snmpd
insserv: warning: current start runlevel(s) (empty) of script `snmpd' overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `snmpd' overrides LSB defaults (0 1 6).
```

Disabling SNMP

Disable SNMP on each individual OVA node with the following commands from an OVA shell prompt as root:

```
systemctl stop snmpd
systemctl disable snmpd
```

Sample output:

```
root@mgmt:~# systemctl stop snmpd
root@mgmt:~# systemctl disable snmpd
snmpd.service is not a native service, redirecting to systemd-sysv-install
Executing /lib/systemd/systemd-sysv-install disable snmpd
insserv: warning: current start runlevel(s) (empty) of script `snmpd' overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `snmpd' overrides LSB defaults (0 1 6).
```

Checking SNMP status

Check whether SNMP is enabled on each individual OVA node with the following commands from an OVA shell prompt:

```
systemctl status snmpd
```

Sample output when SNMP is enabled and running:

```
apicadm@mgmt:~$ systemctl status snmpd
  snmpd.service - LSB: SNMP agents
   Loaded: loaded (/etc/init.d/snmpd; bad; vendor preset: enabled)
   Active: active (running) since Wed 2019-02-06 03:53:02 UTC; 2 days ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 1
   Memory: 6.0M
      CPU: 3min 1.641s
   CGroup: /system.slice/snmpd.service
           └─1366 /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp -g snmp -I -smux mteTrigger mteTriggerConf -p /run/snmpd.pid
...
```

Sample output when SNMP is stopped and disabled:

```
root@mgmt:~# systemctl status snmpd
   snmpd.service - LSB: SNMP agents
   Loaded: loaded (/etc/init.d/snmpd; bad; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:systemd-sysv-generator(8)
...
```

Logging

Configure logging for your deployment on VMware.

See:

- **Configuring remote logging for a VMware deployment**
  Logging collection is required for IBM Support to assist with troubleshooting. You can configure fluent-bit to collect and forward logs to a remote Syslog server.
- **Gathering logs for a VMware environment**
  The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.
- **Changing logging levels**
  You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

Configuring remote logging for a VMware deployment

Logging collection is required for IBM Support to assist with troubleshooting. You can configure fluent-bit to collect and forward logs to a remote Syslog server.

Before you begin

The fluent-bit service is already installed on the appliances and is configured to forward container logs to /var/log/syslog on the local machine.

Important: This article refers to third-party software that IBM does not control. As such, the software may change and this information might become outdated. In addition, the fluent-bit service was tested only for simple container log routing to local and remote Syslog servers as configured in the instructions that follow. Any customization of other fluent-bit settings (for example, to set up parsers or filters) is at your own risk and should be performed with caution by referring to the fluent-bit documentation.

About this task

Remote logs created with fluent-bit use the following format:

```
Mar 1 00:38:44 apicdev1147 calico-node-gqbv_42428a8ade169ad3 2023-03-01 00:38:44.684 [INFO][208] felix/int_dataplane.go 1245: Applying dataplane updates
```

Where:

- `Mar 1 00:38:44` is the time the log arrived on the remote server.
- `apicdev1147` is the hostname of the log source.
- `calico-node-gqbv_42428a8ade169ad3` is a portion of the pod name/container_id where the log is coming from.
- `2023-03-01 00:38:44.684` is the timestamp at which the logged event actually took place within the container.
- The rest of the log is the log message itself.

Logs posted to your local server look different, as in this example:

```
Feb 28 23:47:03 apicdev1147 fluent-bit[2332273]: [2] kube.var.log.containers.kqn2-management-portal-proxy-647b96b89f-pkz5w_default_portal-proxy-74a1723512574fdc23fb9c014b80c7e83aff225b9d44d995736251edd18c49b4.log: [1677628023.503190040, {"_p"=>"F", "log"=>"Tue, 28 Feb 2023 23:47:03 GMT express:router expressInit  : /healthz", "tag"=>"kqn2-management-_74a1723512574fdc", "time"=>"2023-02-28T23:47:03.503190005Z", "stream"=>"stderr"}]
```

On the local log, the whole fluent-bit JSON object is logged, which provides slightly more information (for example, the full file name of the log file) than in the remote log.

To configure remote logging, use the appropriate instructions for your deployed version of API Connect:

- Configuring remote logging
- Updating the remote logging configuration

Configuring remote logging

About this task

The fluent-bit service is already installed on the appliance and is configured to forward container logs to `/var/log/syslog` on the local machine. If you want to change the output destination to a remote Syslog server, complete the following steps:

Procedure

1. Create the /etc/fluent-bit/append-tag.lua file.
   a. Paste the following code into the new file:

```
function append_tag(tag, timestamp, record)
    new_record = record
    local pod_name = string.sub(string.match(tag, "^kube%.var%.log%.containers%.(.+)$"), 1, 16)
    local container_id = string.sub(string.match(tag, "-([^-]+)%.log$"), 1, 16)

    new_record["tag"] = pod_name .. "_" .. container_id
    return 1, timestamp, new_record
end
```

   b. Save and close the file.
2. Create the /etc/fluent-bit/fluent-bit-override.conf file.
   a. Add the following statements to the new file::
   Replace the variables in the `[OUTPUT]` section with values for your deployment:

```
[SERVICE]
        Daemon Off
        Flush 1
        Log_Level info
        Parsers_File parsers.conf
        HTTP_Server On
        HTTP_Listen 0.0.0.0
        HTTP_Port 2021
        Health_Check On

[INPUT]
        Name tail
        Path /var/log/containers/*.log
        multiline.parser docker, cri
        Tag kube.*
        Mem_Buf_Limit 5MB
        Skip_Long_Lines On

[INPUT]
        Name systemd
        Tag host.*
        Systemd_Filter _SYSTEMD_UNIT=kubelet.service
        Read_From_Tail On

[FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
```

```
        K8S-Logging.Parser On
        K8S-Logging.Exclude On

[FILTER]
        Name lua
        Match kube.*
        Script /etc/fluent-bit/append-tag.lua
        call append_tag

[OUTPUT]
        Name syslog
        Match *
        Host <host>
        port 514
        mode udp
        syslog_format rfc5424
        syslog_maxsize 2048
        syslog_hostname_key hostname
        syslog_hostname_preset <hostname of local machine/log source>
        syslog_appname_key tag
        syslog_message_key log
```

Note: The default values shown in the `[OUTPUT]` section can be modified as explained in the fluent-bit documentation.

b. Save and close the file.

3. Run the following command to restart the fluent-bit service so it picks up the configuration changes:

```
systemctl restart appliance-manager && systemctl restart fluent-bit
```

Updating the remote logging configuration

To update the fluent-bit configuration, complete the following steps:

Procedure

1. Run the following command to edit the configuration file:

```
vim /etc/fluent-bit/fluent-bit-override.conf
```

2. Complete your configuration changes, then save and close the file.
3. Run the following command to restart the appliance-manager:

```
systemctl restart appliance-manager
```

Gathering logs for a VMware environment

The `generate_postmortem.sh` script gathers all logs for troubleshooting and diagnostics.

About this task

When you contact IBM Support, logs and VMware environment data are normally required to help diagnosis. Run the `generate_postmortem.sh` script on each API Connect VM to gather all the required logs and data. The data that is gathered by the script includes OS-level information from each VM, so if you have a three replica deployment, it is recommended to run the `generate_postmortem.sh` script on each VM in the cluster.

The logs that are gathered by the `generate_postmortem.sh` script include the logs of the currently running API Connect processes. The logs have a maximum size, and older log messages are not retained when the maximum size is reached. To ensure that the logs gathered cover the time when the problem occurred, it is recommended to reproduce the problem and then immediately gather the logs. Offboarding the logs to an external server is the recommended best practice for long-term storage of log data. See Configuring remote logging for a VMware deployment.

Procedure

1. Connect to the target appliance with SSH:

```
ssh apicadm@<ova appliance hostname>
```

2. Switch to the *root user*:

```
sudo -i
```

3. Download the `generate_postmortem.sh` script. Enter the following command:

```
curl -s -o generate_postmortem.sh https://raw.githubusercontent.com/ibm-apiconnect/v10-postmortem/master/generate_postmortem.sh
```

Note: If your API Connect VMs do not have external internet access, download the `generate_postmortem.sh` script to another computer that has internet access, and copy it to your API Connect VMs.

4. Add execute permissions to the `generate_postmortem.sh` script. Enter the following command:

```
chmod +x generate_postmortem.sh
```

5. Run the postmortem tool by using the following command:

```
./generate_postmortem.sh --ova --pull-appliance-logs
```

6. Optional: Specify either management, portal, or analytics for more detailed logs from that subsystem. Specify `--diagnostic-all` for detailed logs of every subsystem:
   - Portal:

     ```
     ./generate_postmortem.sh --ova --diagnostic-portal --pull-appliance-logs
     ```

   - Management:
     - Download an additional file that gathers management database information: `edb_mustgather.py`:

       ```
       wget https://github.com/ibm-apiconnect/v10-postmortem/blob/master/edb_mustgather.sh
       ```

     - Run `generate_postmortem.sh` with `--diagnostic-manager`:

       ```
       ./generate_postmortem.sh --ova --diagnostic-manager --pull-appliance-logs
       ```

   - Analytics:

     ```
     ./generate_postmortem.sh --ova --diagnostic-analytics --pull-appliance-logs
     ```

   Note: The diagnostic level logs generated with the `--diagnostic-<subsystem name>` argument produce a much larger output file.
7. If you see errors when you run the `generate_postmortem.sh` script, run again with `--debug`:

   ```
   ./generate_postmortem.sh --ova --debug &>debug.log
   ```

   For more documentation on the script, see https://github.com/ibm-apiconnect/v10-postmortem.

Changing logging levels

You can enable logging for entry and exit trace and for large payloads for apim-v2 pods.

About this task

For apim-v2 pods, the default logging settings do not include entry and exit trace and logging of large payloads.

Logging of large payloads is typically needed if you are logging `apim:webhookPayload` or other processes that log large variables.

You can also set the debug level without needing to restart the pods.

You can enable the settings through either the APIM deployment YAML file, the toolkit, or the REST APIs. Note, you must use the toolkit or REST API to set the debug level without restarting the pods.

Procedure

- Updating the settings in the APIM deployment YAML file
  Note: This action causes a pod restart.
  - The DEBUG variable works the same as prior releases.

    ```
    - env:
        - name: DEBUG
          value:
    audit,bhendi:error,bhendi:probe,bhendi:flags,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:
    *
    ```

    To enable entry and exit trace, add `trace:*` to the start of the debug string:

    ```
    - env:
        - name: DEBUG
          value: trace:*,audit,bhendi:error,bhendi:probe,bhendi:flags,bhendi:webhookAudit,bhendi:cassandra-
    transactions,apim:server,apim:error,apim:routes:*,apim:routesc:*,apim:oidc,apim:oidc:*,apim:taskmanager:info
    ```

  - To enable large object logging:

    ```
    - env:
        - name: VELOX_LOG_FULL_PAYLOAD
          value: "true"
    ```

- Updating the settings by using the toolkit
  Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

  ```
  apic log-spec:get
  apic log-spec:update  LOG_SPEC_FILE
  ```

  Where the spec file contains:

  ```
  {"specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error", "large_objects": true}
  ```

- Updating the settings by using the REST API
  Note: This action does not cause a pod restart. Also, this action is for a single APIM pod environment only.

  ```
  GET /cloud/api/log-spec
  PUT /cloud/api/log-spec
  ```

  For example, to set debug level and large objects logging:

```
curl -k https://172.16.140.212:3003/api/cloud/log-spec -X PUT -H "Authorization: Bearer $BEARER"
    -H "Accept: application/json" -d '{"specification":
"apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error","large_objects": true}'
    -H "Content-Type: application/json"
```

The command responds with:

```
{
    "specification": "apim:routes:log_spec,audit,bhendi:error,apim:server,apim:error",
    "large_objects": true,
    "message": "Successfully changed the log specification on all apim-v2 pods. Success on IP(s): 172.16.140.212,
172.16.140.213"
}
```

Note: The response might include the following warning, which can be ignored:

```
WARNING: Could not change the log specification on all apim-v2 pods. Success on IP(s): 4.5.6.7  Failed on IP(s): 1.2.3.4
```

To view the REST APIs for logging, go to API Connect REST APIs, and select IBM API Connect Platform - Cloud Management API 2.0.0 reference > Resource: Log Spec.

Troubleshooting

Troubleshoot your deployment on VMware.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
See:

- **Recreating a calico pod that is in the CrashLoopBackOff state**
  If you complete an upgrade and find that one or more of the calico pods are in the `CrashLoopBackOff` state, you can delete the pod to recreate it.
- **Running a filesystem check on a VMware root partition**
  You can run a filesystem check on the appliance root filesystem at boot time.
- **Troubleshooting the management database V10.0.6.0**
  You can troubleshoot the API Connect management database
- **Increasing the PostgreSQL archive_timeout on VMware**
  Increase the `archive_timeout` to `3600` (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.
- **Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment**
  In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.
- **Troubleshooting analytics on VMware**
  If your API Connect analytics subsystem encounters problems, see the analytics troubleshooting topics.

Recreating a calico pod that is in the `CrashLoopBackOff` state

If you complete an upgrade and find that one or more of the calico pods are in the `CrashLoopBackOff` state, you can delete the pod to recreate it.

Complete the following steps:

1. Get the names of the calico pods:

   ```
   kubectl -n kube-system get pods | grep calico
   ```

2. Recreate each pod by deleting it with the following command:

   ```
   kubectl -n kube-system delete pod <calico_pod_name>
   ```

Running a filesystem check on a VMware root partition

You can run a filesystem check on the appliance root filesystem at boot time.

About this task

If the appliance root filesystem is mounted read-only because it is corrupt, use the following steps to run **fsck** on root partition:

Procedure

1. Access to the VMware console for the Appliance, and after causing it to reboot, make sure to bring up a console to the appliance (as needed to give focus to the window by clicking the mouse into it), and press the `Esc` key repeatedly.
   You will see a boot screen similar to the following:

   ```
   GNU GRUB v2.02
   .
   ```

```
.
- Ubuntu
- Advanced options for Ubuntu
.
.
.
Press Enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line
```

2. Press **e** to edit the commands before booting, and add the lines `fsck.mode=force` and `fsck.repair=yes`.
   For example, the boot screen contents resemble:

```
GNU GRUB v2.02
.
.
set params 'Ubuntu'
    fsck.mode=force
    fsck.repair=yes
.
.
.
Press Ctrl-x or F10 to boot
.
```

3. Press F10 to boot.
4. The following command should show that **fsck** was run recently:

```
root@ip-230:~# sudo tune2fs -l /dev/sda1 | grep "Last checked"
Last checked:             Wed Jul  1 19:32:25 2020
```

Troubleshooting the management database V10.0.6.0

You can troubleshoot the API Connect management database

Database Replica Pods stuck in Unknown or Pending state

In certain scenarios, a postgres replica pod may not recover to a healthy state when a restore completes, a node outage occurs, or after a fresh install or upgrade. In these cases, a postgres pod remains in a **`Unknown`** or a **`Pending`** state after a number of minutes. The pod fail to get into a **`Running`** state.

This situation occurs when the replicas do not initialize properly. You can use the **`patronictl reinit`** command to reinitialize the replica. Note that this command syncs the replica's volume data from the current Primary pod.

Use the following steps to get the pod back into a working state:

1. SSH into the appliance as **`root`**.
2. Exec onto the failing pod:

```
kubectl exec -it <postgres_replica_pod_name> -n <namespace> -- bash
```

3. List the cluster members:

```
patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
|                         Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b   | 172.16.172.244 |        | start failed |    |   unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
```

   In the example shown above **`fxpk-management-01191b80-postgres-586f899fdf-6s25b`** is not in running state.

   Note the **`clusterName`** and **`replicaName`** which are not up:

   - **`clusterName`** - **`fxpk-management-01191b80-postgres`**
   - **`replicaName`** - **`fxpk-management-01191b80-postgres-586f899fdf-6s25b`**

4. Run:

```
patronictl reinit <clusterName> <replicaName-which-is-not-running>
```

   Example:

```
patronictl reinit fxpk-management-01191b80-postgres fxpk-management-01191b80-postgres-586f899fdf-6s25b
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+--------------+----+-----------+
|                         Member                          |      Host      |  Role  |    State     | TL | Lag in MB |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b   | 172.16.172.244 |        | start failed |    |   unknown |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader |   running    |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        |   running    |  3 |         0 |
+---------------------------------------------------------+----------------+--------+--------------+----+-----------+
Are you sure you want to reinitialize members fxpk-management-01191b80-postgres-586f899fdf-6s25b? [y/N]: y
Success: reinitialize for member fxpk-management-01191b80-postgres-586f899fdf-6s25b
```

5. Run `patronictl list` again.
   You may also observe that the replica is on a different Timeline (TL) and possibly have a Lag in MB. It may take a few minutes for the pod to switch onto the same TL as the others and the Lag should slowly go to 0.

   For example:

```
bash-4.2$ patronictl list
+ Cluster: fxpk-management-01191b80-postgres (6893134118851096752) --------+--------+---------+----+-----------+
|                          Member                          |      Host      |  Role  |  State  | TL | Lag in MB |
+----------------------------------------------------------+----------------+--------+---------+----+-----------+
|    fxpk-management-01191b80-postgres-586f899fdf-6s25b    | 172.16.172.244 |        | running |  1 |     23360 |
| fxpk-management-01191b80-postgres-rkww-795665698f-4rh4s | 172.16.148.51  | Leader | running |  3 |           |
|  fxpk-management-01191b80-postgres-uvag-9475f7c5f-qr84m |  172.16.53.68  |        | running |  3 |         0 |
+----------------------------------------------------------+----------------+--------+---------+----+-----------+
```

6. The pod that previously was in an `Unknown` or `Pending state` or `(0/1) Running` state is now in `(1/1) Running` state.

Increasing the PostgreSQL `archive_timeout` on VMware

Increase the **archive_timeout** to **3600** (seconds) so that accumulated wal files do not cause a disk-space issue during a network outage.

About this task

PostgreSQL sends its wal archives to the backup location (S3 or local PVC) whenever the configured **archive_timeout** period elapses. By default, the **archive_timeout** is set to 60 seconds. During a network outage, the files cannot be transferred and instead accumulate on the current server, and could cause a disk-space issue. Increasing the **archive_timeout** value helps to mitigate this issue by creating archive files less frequently.

For information on tracking the Postgres disk space, see Monitoring Postgres disk usage on VMware.

Complete the following steps to increase the PostgreSQL **archive_timeout** value.

Procedure

1. SSH into the server by completing the following steps:
   a. Run the following command to connect as the API Connect administrator, replacing **ip_address** with the appropriate IP address:
      `ssh ip_address -l apicadm`
   b. When prompted, select Yes to continue connecting.
   c. When you are connected, run the following command to receive the necessary permissions for working directly on the appliance:
      `sudo -i`
2. Get the name of the **pgha** config map by running the following command:

   ```
   kubectl -n <namespace> get cm -l pgha-config='true'
   ```

   The response looks like the following example:

   ```
   NAME                                                DATA   AGE
   apis-produ-d18f3f0-0d18f3f0-postgres-pgha-config    5      22d
   ```

3. Edit the config map by running the following command:

   ```
   kubectl -n <namespace> edit configmap <config-map-name>
   ```

   In the command, replace ***<config-map-name>*** with the **NAME** returned in step 1.

4. Change the **archive_timeout** value to **3600**.
   The value must be an integer, and represents the number of seconds that must elapse before the archive file is generated.

5. Save the file and exit the editor.
   The configuration is updated automatically; archives will be generated and pushed every **3600** seconds.

Dynamically re-registering and reconfiguring a Gateway service in a VMware deployment

In API Connect, Gateway services do not persist their configuration settings by default. Instead, the master configuration is stored on the Management server and the *Dynamic Reregistration and Reconfiguration* (DRR) mechanism resynchronizes configuration data when needed. The DRR process is used when proper High Availability/Disaster Recovery (HA/DR) is not configured, or if a manual resynchronization is required.

If a Gateway service is not configured properly for resiliency and is restarted, the gateways in the Gateway service will lose the configuration from the Management server. Configuration data from the Management server is maintained on the gateway service according to the gateway peering configuration on the gateways.

Preventing the loss of configuration data

For high availability in production environments, use a minimum of three gateways in the Gateway service. In non-production environments, if a single gateway is used, then availability can be improved by ensuring the persistence setting on the gateway peering configuration is set to **local://** or **RAID**; these settings apply to virtual and physical appliances only.

Forcing re-population of the configuration data

To repopulate the configuration data, complete the steps in the following section to trigger an outage-less DRR.

Triggering an outage-less DRR

To force a Dynamic Reregistration and Reconfiguration across a peer group from within the gateway service without requiring a restart, complete the following steps using the CLI. This process flushes the primary peering object configured for the apic-gw-service. This task can only be performed with CLI, and requires the default admin ID for running the **diag** command.

1. Use **ssh** to connect to gateway service.
2. Switch to the API Connect application:

   ```
   switch <APIC_APP_DOMAIN>
   ```

3. Show the current gateway-peering-status:

   ```
   show gateway-peering-status
   ```

4. If the server to which the ssh connection was made is not primary for the apic-gw-service gateway-peering object, force this server to become primary:

   ```
   config; gateway-peering-switch-primary <gateway-peering-name-for-apic-gw-service>;
   ```

5. Flush the apic-gw-service gateway-peering object:

   ```
   diag; gateway-peering-flush <gateway-peering-name-for-apic-gw-service>; exit
   ```

6. Disable and then re-enable the apic-gw-service object for every member of the gateway service:

   ```
   config; apic-gw-service; admin-state disabled; exit
   apic-gw-service; admin-state enabled; exit; exit
   ```

7. Confirm that the apic-gw-service was flushed and is now waiting for gateway service registration:
   Look in the debug log target for the apic-gw-service, located in logtemp:/// and check for a message similar to the following example:

   ```
   20200618T232339.332Z [0x88e000d7][apic-gw-service][notice]
   apic-gw-service(default): tid(2653): Waiting for gateway service registration.
   ```

   The DRR will be triggered by the next arrival of a webhook event. The Management server sends a heartbeat to the gateway at five-minute intervals, prompting the gateway to check whether it has lost its configuration and if so, trigger a DRR. In addition, if the Management server has previously marked a Catalog or cloud as being unavailable for a particular gateway service, a successful heartbeat triggers synchronization for that Catalog or cloud on that gateway service.

8. (Optional) Force a BAU webhook event to be sent from API Manager to trigger the DRR:
   If you do not want to wait for the Management server to send a heartbeat, you can trigger the DRR manually by completing the following steps:

   a. Open the API Manager.
   b. Open the Catalog.
   c. Click Settings > Edit.
   d. Update the Summary field with some text so that it is modified.
   e. Click Save.

When the event is received, the DRR is initiated.

Troubleshooting analytics on VMware

If your API Connect analytics subsystem encounters problems, see the analytics troubleshooting topics.

For problems with the analytics subsystem, see Troubleshooting analytics.

Migrating from IBM API Connect v5 Public Cloud to v10.0.6x

You cannot migrate direct to IBM® API Connect 10.0.6.0 or later from API Connect v5 Public Cloud. You must migrate to version 10.0.5.x or later, and then upgrade to 10.0.6.0 or later.

Restriction: A direct migration from API Connect v5 Public Cloud to API Connect 10.0.6.0 or later is not supported. You must migrate to 10.0.5.x as an interim step, and then upgrade to 10.0.6.0 or later if required. For information about migrating to 10.0.5.x, see Migrating from API Connect v5 Public Cloud in the IBM documentation for IBM API Connect 10.0.5.x.

Migrating a version 5 deployment to version 10.0.x

See the following note for information about migrating from API Connect 5.0.x.

Restriction: A direct migration from API Connect 5.0.x to API Connect 10.0.6.0 or later is not supported. You must migrate to 10.0.5.x as an interim step, and then upgrade to 10.0.6.0 or later if required. For information about migrating to 10.0.5.x, see Migrating a version 5 deployment in the IBM documentation for IBM® API Connect 10.0.5.x.

Migrating v5-compatible APIs to API Gateway

Use the migration utility to automate the migration of v5-compatible APIs to DataPower API Gateway APIs.

About this task

If your deployment includes v5-compatible APIs, they can only be used with the DataPower v5-compatible Gateway (also known as the v5c gateway). To use the v5-compatible APIs with the DataPower API Gateway, you must migrate the APIs to the newer format.

The migration process involves the following tasks:

1. Install and configure the required version of API Connect and migration tools.
2. Plan and test your migration to ensure all tools work as intended.
3. Use the API Connect operations tool (`apicops`) to extract API data from your v10.0.7.0 deployment.
4. Use the API Connect migration utility (AMU) to unpack the data into YML files for porting to the newer format.
5. Review the unpacked data and port the v5-compatible APIs to the format required by the DataPower API Gateway.
6. Use the AMU to load the ported APIs back into your v10.0.7.0 deployment.

- **Preparing the environment**
  Install all of the required applications before attempting to migrate v5-compatible APIs to the DataPower API Gateway.
- **Planning your API migration**
  Plan how you will migrate your v5-compatible APIs and then test the process.
- **Downloading and extracting the data to migrate**
  Use `apicops` to download data from API Connect, and then use the AMU to extract the data from the downloaded archive.
- **Converting v5-compatible API definitions**
  To convert v5-compatible APIs to the format required by the DataPower API Gateway, you must map all v5-compatible gateway services to DataPower API Gateway services, optionally specify any configuration settings to be used in the conversion, and use the AMU to port the API definitions.
- **Pushing custom policies and gateway extensions to the DataPower API Gateway**
  Use the AMU to push the Cloud Manager admin organization's data to the Management server, which pushes custom policies and gateway extensions to your DataPower API Gateway Services.
- **Pushing provider organization data**
  Use the AMU to push the provider organization data to the Management server.

Preparing the environment

Install all of the required applications before attempting to migrate v5-compatible APIs to the DataPower API Gateway.

About this task

Migrating v5-compatible APIs to the DataPower API Gateway requires the following applications:

- API Connect v10.0.7.0
- API Connect operations (`apicops`) tool, latest v10 release
- API Connect migration utility (AMU), release v10.0.7.0-Rx
- Node version 18.19.1 or later

Complete the following steps to set up your environment with the required applications:

Procedure

1. Upgrade to API Connect v10.0.7.0.
   Your API Connect deployment must be at 10.0.7.0. Refer to the upgrade instructions in the Installing and maintaining IBM API Connect section.

2. Download and configure the latest release of the `apicops` tool.
   Download the `apicops` binary for your operating system from GitHub at latest release and rename the file to apicops.

   Review the associated README for requirements and instructions for using the tool.

3. Download the migration utility v10.0.7.0-Rx.
   The API Connect migration utility (AMU) is available from IBM Fix Central. On Fix Central, set the Installed version to 10.0.7.0, and search for the text "migration" to locate the AMU.

4. Install Node version 18.19.1 or later.

What to do next

After you have installed the required applications for your environment, complete the following tasks to migrate your v5c-compatible APIs.

Planning your API migration

Plan how you will migrate your v5-compatible APIs and then test the process.

About this task

Plan the migration process, migrate a small set of APIs, and then test the results. You can refine settings and repeat the process as needed.

Procedure

1. Review all of the migration topics to ensure you understand the requirements, the general process, and important considerations.
2. Plan how you will migrate your v5-compatible APIs to the DataPower Gateway.
   Make sure you know exactly which APIs you want to migrate, and in which provider organizations and catalogs they are stored.
3. Migrate a proof-of-concept deployment.
   Migrate 10 to 20 of your most commonly used v5-compatible APIs to ensure that your migration tools are configured correctly and the migration process works as intended. Plan to run several test iterations of migration.

   You can use the migration utility's **`--dry-run`** option to perform iterations of the migration push without actually uploading data to API Connect. The test runs offer an opportunity to test for errors and warnings that need to be resolved before you perform the actual data push.

   You can use the migration utility's **`--move-subscriptions`** parameter to move subscriptions during the push operation that uploads converted data back into API Connect.
4. Test the APIs that you migrated as proof-of-concept.
   Testing the migrated APIs helps you to determine which migration utility settings you need for successfully converting the v5-compatible APIs to DataPower API Gateway. If needed, you can move or delete the results, modify the configuration settings, and run the migration again. When you rerun the migration **`port`** command using the same target folder for output, the utility skips artifacts that have already been converted to DataPower API Gateway.

Downloading and extracting the data to migrate

Use **`apicops`** to download data from API Connect, and then use the AMU to extract the data from the downloaded archive.

Procedure

1. Use **`apicops`** to extract and download data from API Connect as a .pgdmp archive file.
   Run the following command:

   `apicops postgres:pg_dump -e <archive_name>.pgdmp`

   where **`<archive_name>.pgdmp`** is a name that you create for the archive file that will contain the downloaded data; for example, v10_backup.pgdmp.
2. Copy the .pgdmp archive file to the location where you installed the AMU, storing it in the same directory as the AMU.
3. Use the AMU to unpack the archive stored in the .pgdmp archive file.
   Run the following command in the same directory where you stored the downloaded .pgdmp archive file (see Table 1 for available the parameters):

   On Windows: open a command window with Administrator privileges, and omit the "./" from the command.

   `./apicm archive:unpack-v10 <archive_name>.pgdmp`

   The first time you use the AMU, you are prompted to accept the license. You must accept the license to continue.

   Table 1. AMU **`archive:unpack-v10`** parameters

Parameter	Description
<archive_name>.pgdmp	Required. The .pgdmp archive file created containing the data downloaded from API Connect.
`--provider-orgs`	Optional. A comma-separated list containing one or more provider organization names. Use this parameter to unpack only the data for the specified provider organizations instead of unpacking the entire archive. Example: **`./apicm archive:unpack-v10 <archive_name>.pgdmp --provider-orgs=<pOrg1_name>,<pOrg2_name>,<pOrg3_name>`**

   When the command completes, the data is unpacked to the /cloud-v10 directory, which is located in the same directory as the AMU, and is populated with YAML files representing the configuration objects in the .pgdmp file.

   Important: The unpack directory must be named /cloud-v10. Do not rename the directory.

Converting v5-compatible API definitions

To convert v5-compatible APIs to the format required by the DataPower API Gateway, you must map all v5-compatible gateway services to DataPower API Gateway services, optionally specify any configuration settings to be used in the conversion, and use the AMU to port the API definitions.

Procedure

1. Map your v5-compatible Gateway Services to DataPower API Gateway services.
   You must map your v5-compatible Gateway Services (the v5c gateway) to DataPower API Gateway services so that the AMU command has a target for migrating custom policies and gateway extensions, and can migrate the configured gateway services for each catalog and space.

   a. In the unpacked data, locate the gateway services.
      Each of the gateway services in stored in cloud-v10/admin-org/availability-zones/*<az_name>*/gateway-services/.

   b. For each v5-compatible gateway service, create a mapping.yml file that contains the name of the DataPower API Gateway Service you want to map to.
      Create the mapping.yml file directly in the directory containing the v5-compatible gateway service. The file must contain 3 statements:
      - The artifact type, which is always **`gateway-service:`**

- The name of the DataPower API Gateway service (not the Title or ID)
- The gateway type for that gateway service, which is always **datapower-api-gateway** for converting to DataPower API Gateway service

For example, the following mapping.yml file maps the v5-compatible gateway service to the DataPower API Gateway service named "gateway-10-1" and specifies a target gateway of type **datapower-api-gateway**:

```
gateway-service:
  name: gateway-10-1
  gateway-type: datapower-api-gateway
```

If you have several v5-compatible gateway services and you want to map each of them to the same DataPower API Gateway service, copy the mapping.yml file into the directory for each of those v5-compatible gateway services.

Make sure to map all of the v5-compatible gateway services to DataPower API Gateway services before attempting to port the APIs in the next step.

2. Optional: Create a configuration file that defines settings for migrating the v5-compatible APIs.
In the directory where the AMU is stored, create a file called **config.yml** and use it to specify configuration settings (see Table 1 for the configuration settings).

Example **config.yml** file:

```
custom-policies-scope: global
emulate-v4-plan-rate-limit: false
deploy-policies:
  - proxy_1.5.0
  - redact_1.5.0
  - invoke_1.5.0
  - xslt_2.0.0
  - gatewayscript_1.0.0
  - switch_2.0.0
rewrite-apis:
  enable-api-logging: false
  proxy-policy: proxy_1.5.0
  redact-policy: redact_1.5.0
  invoke-policy: invoke_1.5.0
  xslt-policy: xslt_2.0.0
  gatewayscript-policy: gatewayscript_1.0.0
  if-policy: switch_2.0.0
  switch-policy: switch_2.0.0
  optimize-gws: false
  no-retitle: true
```

If you omit the config.yml file, you can optionally specify configuration settings as parameters on the command line when you port the APIs in the next step.

Table 1. Migration configuration settings

Parameter	Valid values	DataPower version required	Description
activity-log-policy	**activity-log_1.5.0, set-variable_2.0.0**	DP 10.0.4.0+	Defines how to process the v5 activity-log policy when the migration utility rewrites the API assembly. The default value is **activity-log_1.5.0**, which changes the activity-log version to 1.5.0. When set to **set-variable_2.0.0**, the migration utility sets context variables based upon the value of the activity-log content and error-content configuration values.
chunked-uploads	**true / false** The default value is **false**.	For **invoke_1.5.0**: 10.0.1.1+, 10.0.2.0+ For **invoke_2.0.0**: 10.0.0.0+	If an **invoke_1.5.0** or **proxy_1.5.0** policy exists in an assembly in the API, then set **chunked-uploads** to **true** to enable chunked encoding for HTTP 1.1 requests to the target server. If an **invoke_2.0.0** policy exists in an assembly after the migration, then set **chunked-uploads** to **false** explicitly for the policy, as the default value for **chunked-uploads** in API Gateway is **true**.
client-id-header-override	String. The string cannot contain spaces. Follow the guidelines for **apiKey name:**. No default value.	10.0.0.0+	Override the existing **X-IBM-Client_Id** name value with the new value. In API Gateway, **x-key-type** makes it possible to define custom naming for **client_id** and **client_secret**. Example: **--client-id-header-override="acme-foo"** In this example, the **securityDefinitions** property is changed from: **foo:** **type: apiKey** **name: X-IBM-Client-Id** **in: header** to the following: **foo:** **type: apiKey** **name: acme-foo** **in: header** **x-key-type: client_id** Calls to this API must supply **acme-foo: <client-id>** in the request header instead of the original **X-IBM-Client-Id: <client-id>**.

Parameter	Valid values	DataPower version required	Description
client-id-query-override	String. The string cannot contain spaces. Follow the guidelines for **apiKey name:**. No default value.	10.0.0.0+	Override the existing **client_id** name value with the new value. In API Gateway, **x-key-type** makes it possible to define custom naming for **client_id** and **client_secret**. Example: **--client-id-query-override="acme-foo"** In this example, the **securityDefinitions** property is changed from: **foo:** **type: apiKey** **name: client_id** **in: query** to the following: **foo:** **type: apiKey** **name: acme-foo** **in: query** **x-key-type: client_id** Calls to this API must supply **?acme-foo=<client-id>** in the query parameter instead of the original **?client_id=<client-id>**.
client-secret-header-override	String. The string cannot contain spaces. Follow the guidelines for **apiKey name:**. No default value.	10.0.0.0+	Override the existing **X-IBM-Client-Secret** name value with the new value. In API Gateway, **x-key-type** makes it possible to define custom naming for **client_id** and **client_secret**. Example: **--client-secret-header-override="acme-foo"** In this example, the **securityDefinitions** property is changed from: **foo:** **type: apiKey** **name: X-IBM-Client-Secret** **in: header** to the following: **foo:** **type: apiKey** **name: acme-foo** **in: header** **x-key-type: client_secret** Calls to this API must supply **acme-foo: <client-secret>** in the request header instead of the original **X-IBM-Client-Secret: <client-secret>**.
client-secret-query-override	String. The string cannot contain spaces. Follow the guidelines for **apiKey name:**. No default value.	10.0.0.0+	Override the existing **client_secret** name value with the new value. In API Gateway, **x-key-type** makes it possible to define custom naming for **client_id** and **client_secret**. Example: **--client-secret-query-override="acme-foo"** In this example, the **securityDefinitions** property is changed from: **foo:** **type: apiKey** **name: client_secret** **in: query** to the following: **foo:** **type: apiKey** **name: acme-foo** **in: query** **x-key-type: client_secret** Calls to this API must supply **?acme-foo=<client-secret>** in the query parameter instead of the original **?client_secret=<client-secret>**.
copy-id-headers-to-message	**true / false** The default value is **false**.	10.0.1.1+, 10.0.2.0+	When set to **true**, the **client-id** header is copied from **request.headers** to **message.headers** to be passed to the backend server. Set this parameter to **true** if your invoke backend services expect the **client-id** to be passed through.

Parameter	Valid values	DataPower version required	Description
custom-policies-scope	**catalog / global** The default value is **catalog**.	10.0.0.0+	When set to **catalog**, the original v5 or v5-compatible catalog that contains the custom policy is staged for pushing to the target catalog and organization in API Gateway. When set to **global**, the custom policy is staged for pushing to API Gateway and is visible to all organizations and catalogs within that gateway. **archive:port-to-apigw-v10** sets up the custom policies for pushing to API Gateway, but does not do the pushing. If **global** scope is used, **archive:push-v10** of the admin organization will push all custom policies associated with the configured gateways. If **catalog** scope is used, then the **archive:push-v10** of the admin organization will initialize the custom policies on the gateway. A subsequent **archive:push-v10** of the provider organization is needed to advertise the custom policies in the catalogs contained within the provider organization. Note: Potential conflicts between custom policies of the same name and version are resolved according to the following rules: • When set to **global** and two policies under the same gateway service have the same name, version, and md5 hash value, then the duplicates are discarded and a message is generated. • When set to **global** and two policies under the same gateway service have the same name, version, and a differing md5 hash value, then global scope cannot be applied. Catalog scope is used and a warning is generated. • When set to **catalog**, **<orgname>_<catname>_** is appended to the zip filename to maintain uniqueness.
deploy-policies	Any combination of the following values: **gatewayscript_1.0.0**,**if_1.5.0**, **invoke_1.5.0**, **proxy_1.5.0**,**redact_1.5.0**, **switch_1.5.0**, **validate-usernametoken_1.0.0**, **xslt_1.0.0** The default values are: **if_1.5.0**, **invoke_1.5.0**, **proxy_1.5.0**, **redact_1.5.0**, **switch_1.5.0**, **validate-usernametoken_1.0.0**, **xslt_1.0.0**	For **invoke_1.5.0**, **proxy_1.5.0**, **gatewayscript_1.0.0**, and **xslt_1.0.0**: 10.0.0.0+ For **if_1.5.0** and **switch_1.5.0**: 10.0.1.1+, 10.0.2.0+ For **redact_1.5.0**: 10.0.3.0+ For **validate-usernametoken_1.0.0**: 10.0.1.3+, 10.0.2.0+	Defines which backwards-compatible policies should be advertised by DataPower to API Manager. Note: Any legacy policies that are specified in the **config.yml** file or by using the **apicm** command must also be specified by using **deploy-policies**. If these policies are not specified by using **deploy-policies**, an error is returned.
emulate-v4-plan-rate-limit	**true / false** The default value is **false**.	10.0.1.2+, 10.0.2.0+	Determines whether all APIs emulate the v4 plan rate limit. If set to **true**, then API operations do not share the rate limit. This setting matches the behavior of the **x-ibm-gateway-emulate-v4-plan-rate-limit** v5 option.
enable-api-logging	**true / false** The default value is **true**.	10.0.0.0+	Log all changes that the migration utility makes when rewriting APIs. These changes are logged as comments in the API YAML file.
enforce-required-params	**true / false** The default value is **false**.	10.0.1.1+, 10.0.2.0+	When set to **true**, required parameters are enforced, which is the default behavior in API Gateway. When set to **false**, required parameters are not enforced, even if the required parameter checkbox selected.
gatewayscript-policy	One of the following values: **gatewayscript_1.0.0**, **gatewayscript_2.0.0** The default value is **gatewayscript_2.0.0**.	10.0.0.0+	Defines which gatewayScript policy the migration utility uses when rewriting the API assembly. **gatewayscript_1.0.0**: A user-defined policy created by the migration utility. This policy uses the v5 MPGW processing rule. Use **gatewayscript_1.0.0** if the gatewayScript policy is dependent on v5 MPGW capabilities, such as load balancer groups or XML Manager settings. **gatewayscript_2.0.0**: The migration utility rewrites the existing API to gatewayScript 2.0.0. This policy supports the **header-metadata** module for compatibility with the v5-compatible gateway. Specifying **gatewayscript_2.0.0** can cause migration errors or run-time errors. Use **gatewayscript_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.
if-policy	One of the following values: **if_1.5.0**, **switch_2.0.0** The default value is **if_1.5.0**.	For **if_1.5.0**: 10.0.1.1+, 10.0.2.0+ For **switch_2.0.0**: 10.0.0.0+	Defines which **if** policy the migration utility uses when rewriting the API assembly. **if_1.5.0**: The migration utility rewrites the existing API to if 1.5.0. **switch_2.0.0**: The migration utility rewrites the existing API to switch 2.0.0. Specifying **switch_2.0.0** can cause migration errors or run-time errors. Use **switch_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.

Parameter	Valid values	DataPower version required	Description
invoke-policy	One of the following values: **invoke_1.5.0**, **invoke_2.0.0** The default value is **invoke_1.5.0**.	10.0.0.0+	Defines which invoke policy the migration utility uses when rewriting the API assembly. **invoke_1.5.0**: The migration utility rewrites the existing API to invoke 1.5.0. **invoke_2.0.0**: The migration utility rewrites the existing API to invoke 2.0.0. Specifying **invoke_2.0.0** can cause migration errors or run-time errors. Use **invoke_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.
no-retitle	**true / false** The default value is **false**.	10.0.0.0+	Defines whether to append the porting suffix to the **title** of the migrated API, product, or OAuth provider. Set this option to **true** if you want the **title** of the migrated API, product, or OAuth provider to match the original **title** in the v5-compatible YAML file.
optimize-gws	**true / false** The default value is **false**.	10.0.0.0+	When set to **true**, gatewayScript policies are modified to map v5 functions to API Gateway functions. Setting **optimize-gws** to **true** is valid only for the **gatewayscript_2.0.0** policy. If it is set to **true** for gatewayscript_1.0.0, an error is generated. Note: Setting **optimize-gws** to **true** can result in performance enhancements. However, the modifications to gatewayScript policies by the migration utility can cause problems. When using this setting, review all gatewayScript policy modifications after migration.
porting-suffix	A string containing alphanumeric, dash, underscore and dot characters. The default value is **-apigw**.	10.0.0.0+	Defines the suffix to be used when renaming APIs, Products, OAuth Providers, and other ported artifacts. The suffix is appended to the version on APIs, Products and other ported artifacts that have versions, and onto the name for OAuth Providers and other ported artifacts that do not have versions. If **porting-suffix** is not specified then a value of **-apigw** will be used. The porting suffix can contain alphanumeric, dash, underscore and dot characters. Prior to AMU 10.0.5.4 the porting suffix could not be overridden and **-apigw** was appended to the name for all ported artifacts.
proxy-policy	One of the following values: **proxy_1.5.0**, **invoke_2.0.0** The default value is **proxy_1.5.0**.	10.0.0.0+	Defines which proxy policy the migration utility uses when rewriting the API assembly. **proxy_1.5.0**: The migration utility rewrites the existing API to **proxy 1.5.0**. **invoke_2.0.0**: The migration utility converts the proxy policy to invoke 2.0.0. Specifying **invoke_2.0.0** can cause migration errors or run-time errors. Use **invoke_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.
redact-policy	One of the following values: **redact_1.5.0**, **redact_2.0.0** The default value is **redact_1.5.0**.	10.0.3.0+	Defines which redact policy the migration utility uses when rewriting the API assembly. **redact_1.5.0**: The migration utility rewrites the existing API to **redact 1.5.0**. **redact_2.0.0**: The migration utility rewrites the existing API to **redact 2.0.0**. Specifying **redact_2.0.0** can cause migration errors or run-time errors, including changes to what is redacted, particularly with redactions meant for JSON payloads. Use **redact_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.
repair-wsdl-apis	**true / false** The default value is **true**.	10.0.1.1+, 10.0.2.0+	If an API ported from V5 is created from WSDL, the WSDL is read and checked for accuracy. If **repair-wsdl-apis** is set to **true**, then the API and WSDL are automatically repaired. For example, if the WSDL ZIP file contains unnecessary files, they are automatically removed if **repair-wsdl-apis** is **true**.
return-v5-responses	**true / false** The default value is **false**.	10.0.1.1+, 10.0.2.0+	When set to **true**, OAuth-related messages are printed in the v5 response format. When set to **false**, OAuth-related response messages are printed in API Gateway response format.
switch-policy	**switch_1.5.0**, **switch_2.0.0** The default value is **switch_1.5.0**.	For **switch_1.5.0**: 10.0.1.1+, 10.0.2.0+ For **switch_2.0.0**: 10.0.0.0+	Defines which switch policy the migration utility uses when rewriting the API assembly. **switch_1.5.0**: The migration utility rewrites the existing API to switch 1.5.0. **switch_2.0.0**: The migration utility rewrites the existing API to switch 2.0.0. Specifying **switch_2.0.0** can cause migration errors or run-time errors. Use **switch_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.
use-config-file	**true / false** The default value is **false**.	10.0.0.0+	Use the **config.yml** file that is in the same directory as the **apicm** binary file that is being executed. If this file is used, configuration options cannot be set using the command line. The purpose of the config file is to specify the command inputs given on this page, but in an easy-to-read file format (YAML).
v5-request-headers	**true / false** The default value is **true**.	10.0.1.2+, 10.0.2.0+	When set to **true**, the following v5-compatible request headers are set: **Via**, **X-Global-Transaction-ID**, **X-Client-IP**. When set to **false**, these request headers are not set.
validate-usernametoken-policy	**validate-usernametoken_1.0.0** The default value is empty.	10.0.1.3+, 10.0.2.0+	Defines whether or not to use the **validate-usernametoken** v5 emulation policy when the migration utility rewrites the API assembly. When empty, the migration utility takes no action on the policy.
xslt-policy	One of the following values: **xslt_1.0.0**, **xslt_2.0.0** The default value is **xslt_1.0.0**.	10.0.0.0+	Defines which gatewayscript policy the migration utility uses when rewriting the API assembly. **xslt_1.0.0**: A user-defined policy created by the migration utility. This policy uses the v5 MPGW processing rule. Use **xslt_1.0.0** if the xslt policy is dependent on v5 MPGW capabilities, such as load balancer groups or XML Manager settings. **xslt_2.0.0**: The migration utility rewrites the existing API to xslt 2.0.0. Specifying **xslt_2.0.0** can cause migration errors or run-time errors. Use **xslt_2.0.0** if you are able to review and verify all changes made by the migration utility in the API YAML file.

3. Use the AMU **port** command to convert the v5-compatible APIs to the DataPower API Gateway format.

When you run the **port** command with the AMU, there are 3 options for specifying configuration settings for your APIs:

- Store the settings in the config.yml migration configuration file.
  You must name the configuration file config.yml. Specify the **--use-config-file=true** parameter on the command line when you port the APIs.
- Include the configuration settings on the command line (see Table 2 for the configuration settings).
  Specify the settings as parameters on the command line. If you also specify a configuration file, any configuration settings specified as parameters directly on the command line are ignored.
- Accept the default configuration settings.
  If you don't specify a configuration file and also don't specify any configuration parameters on the command line, the AMU uses the default configuration when porting the APIs.

To port the APIs, run the following command:

On Windows: open a command window with Administrator privileges, and omit the "./" from the command.

```
./apicm archive:port-to-apigw-v10 <path_to_APIs> --enable-api-logging=false --use-config-file=true
```

where **<path_to_APIs>** represents one of the following locations:

- The top-level /cloud-v10 directory containing all of the extracted archive:
  The AMU ports all artifacts within the directory, including APIs, Products, OAuth providers, and custom policies. Running a subsequent **archive:push-v10** command pushes all of the converted artifacts to the DataPower API Gateway. Custom policies and OAuth providers are not migrated if a directory other than /cloud-v10 is specified.
  Attention: Running the **port** command on any subdirectory within the unpacked /cloud-v10 directory could leave the directory in an inconsistent state, and cause the subsequent **archive:push-v10** command to fail.
- A single API file, or a directory that contains only APIs or Products:
  When you are testing your migration settings, you might want to convert just a few APIs or Products to verify the settings. When you use this approach, you cannot use the high-level **archive:push-v10** command to push the artifacts to the DataPower API Gateway. Instead, you must push the ported artifacts into API Manager using either the toolkit or the user interface.

  When testing is complete, you can delete the artifacts from testing and run the **port** command on the top-level /cloud-v10 directory to convert all of the artifacts in one operation, and then use the **archive:push-v10** command to push them all to the DataPower API Gateway in a second operation.

Table 2. Parameters for the **port** command

Parameter	Description
--enable-api-logging=false	Optional. Disable logging. By default, logging is enabled for the conversion and all changes that the migration utility makes when rewriting APIs are logged as comments within the API's YAML file.
--use-config-file=true	Optional. When converting APIs, apply the configuration settings specified in the config.yml file that is stored in the same directory as the AMU.
Parameters from Table 1, Migration configuration settings.	Optional. Follow the format shown in Table 1, and prefix each parameter with the -- symbol and leave a blank space before the next parameter as shown in the example command.

4. Review the ported artifacts.
   The API conversion process annotates the YAML file and logs with information regarding the artifacts that it attempts to convert. For cases where additional information is available or further action is required, the YAML input file and logs contain links to documentation.

   Review the entire draft output YAML file for conformance, performance, and errors. Review the log files for messages about any manual actions that might be needed. For more information, see Reviewing APIs converted to DataPower API Gateway. By default, all changes that the migration utility makes when rewriting APIs are logged as common.

   Because of differences between the programming models of v5-compatible APIs and DataPower API Gateway APIs, the command might not be able to convert all items; you might need to manually convert the failed items. In these cases, messages generated by the **apicm archive:port-to-apigw-v10** command provide links to documentation that provides more details about changes you must make for the items to be compatible with DataPower API Gateway.

Pushing custom policies and gateway extensions to the DataPower API Gateway

Use the AMU to push the Cloud Manager admin organization's data to the Management server, which pushes custom policies and gateway extensions to your DataPower API Gateway Services.

About this task

Custom policies and gateway extensions are stored with the Cloud Manager admin organization data. To push the converted artifacts to the DataPower API Gateway Services, you push the Cloud Manager admin organization to the Management server.

Procedure

1. Log in to the Cloud Manager on the destination v10 Management Server as an Owner of the admin organization.
   Important: You must be logged in as an Owner of the admin organization to be able to migrate Cloud Manager admin organization data.
   On Windows: open a command window with Administrator privileges, and omit the "./" from the command.

   See Table 1 for the login parameters.

   Syntax:

   ```
   ./apicm login --server <cloud_manager_endpoint> --realm admin/<identity_provider> --username admin --password
   admin_password
   ```

Example for standalone deployment:

```
./apicm login --server cloud-admin-ui.mgmt.dev.apic10-stack.example.test --realm admin/default-idp-1 --username admin --password 'myadminpassword'
```

Example for Cloud Pak for Integration:

```
./apicm login --server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm admin/default-idp-1 --username admin --password 'myadminpassword'
```

Table 1. `login` parameters for the Cloud Admin organization

Parameter	Description

`--server <cloud_manager_endpoint>`	Required. The endpoint on the Management server for communication with the Cloud Manager user interface.
	
	Standalone deployment
	Example:
	`cloud-admin-ui.mgmt.dev.apic10-stack.example.test`
	
	The endpoint was specified when the management subsystem was installed. View the `apiconnect_api` definition in the `apiconnect-up.yml` file in the API Connect v10 installation project directory that you created for initial deployment of v10. Examples:
	• VMware: see the `cloud-admin-ui` endpoint in Deployment overview for endpoints.
	• Kubernetes: see the `admin` endpoint in Deployment overview for endpoints and certificates.
	Cloud Pak for Integration
	Syntax:
	`cpd-<APIC_namespace>.<your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance>`
	
	Example:
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`
	
	where `e2e` is *<APIC_namespace>* and `prod` is *<APIC_instance>*.
	
	Note: How to obtain the server URL on Cloud Pak for Integration
	Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using `oc get routes` because Zen modifies the route internally.
	a. Locate the Platform UI panel with entries for the deployment. Identify the row that has a `Type` column entry with the description `API management administration`.
	b. To display the endpoint, hover over the corresponding **prod** entry in the `Name` column for that row. The URL displays at the bottom of the panel. .
	`Name Type`
	`prod API management administration`
	`prod API management`
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/admin`
	c. Remove the trailing `/admin` from the endpoint:
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`

`--realm admin/default-idp-1`	Required. The `apicm login` example shows the expected `--realm` parameter values for a default API Connection administration organization. The default user is `admin` and the identity provider is `default-idp-1`.
	You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):`apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm`For example:
	`apic identity-providers:list --scope admin --server myserver.com --fields title,realm`
	`total_results: 2`
	`results:`
	`- title: Cloud Manager User Registry`
	`realm: admin/default-idp-1`
	`- title: Corporate LDAP user registry`
	`realm: admin/corporate-ldap`
	
	The `title` value indicates which identity provider to use; copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.

`--username admin`	Required. Administrative user with permissions to modify API Connect admin-org. The default user is `admin`.
`--password myadminpassword`	Required. Administrative user's password. Cloud Pak for Integration: When you installed with the top-level CR for CP4I, the password was auto-generated. To obtain the password, see step 5 in Installing API Connect:

2. Push the ported custom policies and gateway extensions to your DataPower API Gateway services.
In the same directory where you ran the `archive:unpack-v10` command, run the following command to push the ported custom policies and gateway extensions to your DataPower API Gateway services in the Cloud Manager administrative org (admin-org).

See the parameters in Table 2 as well as the usage notes that follow the table:

```
./apicm archive:push-v10 <cloud_manager_endpoint> <path_to_admin_org> [optional flags]
```

Example for standalone deployment (optional flags omitted):

```
./apicm archive:push-v10 cloud-admin-ui.mgmt.dev.apic10-stack.example.test cloud-v10/admin-org
```

Example for Cloud Pak for Integration (optional flags omitted):

```
./apicm archive:push-v10 cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud-v10/admin-org
```

Note: If you observe error messages in your log output, you can take troubleshooting steps, and run the migration again. See Migration troubleshooting .

Table 2. Parameters for the `archive:push-v10` command

Parameter	Description
<cloud_manager_endpoint>	Required. The endpoint on the Management server for communication with the Cloud Manager user interface. Standalone deployment Example: **`cloud-admin-ui.mgmt.dev.apic10-stack.example.test`** The endpoint was specified when the management subsystem was installed. View the **`apiconnect_api`** definition in the **`apiconnect-up.yml`** file in the API Connect v10 installation project directory that you created for initial deployment of v10. Examples: • VMware: see the **`cloud-admin-ui`** endpoint in Deployment overview for endpoints. • Kubernetes: see the **`admin`** endpoint in Deployment overview for endpoints and certificates. Cloud Pak for Integration Syntax: ***<cpd-APIC_namespace.your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance>*** Example: **`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`** where **`e2e`** is *<APIC_namespace>* and **`prod`** is *<APIC_instance>*. Note: How to obtain the server parameter on Cloud Pak for Integration Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using **`oc get routes`** because Zen modifies the route internally. a. Locate the Platform UI panel with entries for the deployment. Identify the row that has a Type column entry with the description **`API management administration`** b. To display the endpoint, hover over the entry in the Name column for that row. For example, hover over the **prod** entry, as shown in bold font, and the URL displays at the bottom of the panel. **`Name Type`** **`prod API management administration`** **`prod API management`** **`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/admin`** c. Remove the trailing **`/admin`** from the endpoint: **`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`**
<path_to_admin_org>	Required, the only supported value is cloud-v10/admin-org.
`--dry-run`	Optional. Perform a dry run to test the Admin org migration output. This option creates a project directory similar to cloud-v10, but named .workdir-*<apim_admin_server>*, where you can view how the data will be loaded into your new system. This option also displays the objects that will fail due to pre-existing or missing dependencies. For more information, see the 10.0.5.x Migration Troubleshooting section on apicm command line help.

Usage notes:

- On Windows: open a command window with Administrator privileges, and omit the "./" from the command.
- Run the **`archive:push-v10`** command in the same directory where you ran the **`archive:unpack-v10`** command, and always push the ported data back to the same v10 deployment where you extracted the data.
- The **`login`** command obtains a token that expires after eight hours.
  If the token expires while the **`archive:push-v10`** command is running, the AMU generates errors. In this case, log in again and re-run the **`push`** command. The token is refreshed every time you log in. For more information, see Token expiration .
- An alternative method of viewing endpoints is to run the **`kubectl get ingress`** and get the values from the output. For example, see the values in **bold** in the following sample output:

```
r7c221d8a33-apic-portal-director api.portal.dev.apic10-stack.example.test 80, 443 21h
r7c221d8a33-apic-portal-web portal.dev.apic10-stack.example.test 80, 443 21h
re314d70920-apiconnect-api platform.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-apim-ui apim.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-cm-ui cloud.mgmt.dev.apic10-stack.example.test 80, 443 22h
re314d70920-apiconnect-consumer-api consumer.mgmt.dev.apic10-stack.example.test 80, 443 22h
rfe867a858b-dynamic-gateway-service service.gw.dev.apic10-stack.example.test 80, 443 21h
rfe867a858b-dynamic-gateway-service-gw api.gw.dev.apic10-stack.example.test 80, 443 21h
rfff5043330-analytics-client ac.dev.apic10-stack.example.test 80, 443 21h
rfff5043330-analytics-ingestion ai.dev.apic10-stack.example.test
```

Pushing provider organization data

Use the AMU to push the provider organization data to the Management server.

About this task

Pushing the provider organization data uploads the ported OAuth Providers, draft APIs, draft products, configured gateway services, and published products to the Management server.

Procedure

1. Log in to the API Manager user interface as an Owner or Administrator of the provider organization that you want to migrate.
   If you want to login using an OIDC user registry, see Logging in to a management server with an OIDC registry.

   Important: You cannot migrate a provider organization while you are logged in with the Cloud Manager admin organization. You must be logged in as the Owner or Administrator of the provider organization whose data you will push.
   Syntax:

   ```
   ./apicm login --server <api-manager-ui-endpoint> --realm provider/<identity_provider> --username <org_username> --password <org_password>
   ```

   Example for a standalone deployment:

   ```
   ./apicm login --server api-manager-ui.dev.apic10-stack.example.test --realm provider/default-idp-2 --username org_username@example.com --password myorgpassword
   ```

   Example for Cloud Pak for Integration:

   ```
   ./apicm login --server cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod --realm provider/default-idp-2 --username org_username@ex
   ```

   Table 1. `login` parameters for Provider organizations

Parameters	Description

<api-manager-ui-endpoint>	Required. The API Manager URL endpoint on the Management server for communication with the API Manager user interface.
	
	Standalone deployment
	Example:
	`api-manager-ui.dev.apic10-stack.example.test`
	The endpoint was specified when the Management subsystem was installed. To verify your endpoint setting, do one of the following:
	• View the `apiconnect-apim-ui` definition in the `apiconnect-up.yml` file in the API Connect installation project directory.
	• In the Cloud Manager UI, select Settings > Endpoints and view API Manager URL.
	Cloud Pak for Integration
	Syntax:
	`cpd-<APIC_namespace>.<your-company.domain.com>/integration/apis/<APIC_namespace>/<APIC_instance>`
	Example:
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`
	where `e2e` is *<APIC_namespace>* and `prod` is *<APIC_instance>*.
	Note: How to obtain the server URL on Cloud Pak for Integration
	Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using `oc get routes` because Zen modifies the route internally.
	a. Locate the Platform UI panel with entries for the deployment. Identify the row that has a `Type` column entry with the description `API management`
	b. To display the endpoint, hover over the corresponding **prod** entry in the `Name` column for that row. The URL displays at the bottom of the panel.
	`Name Type`
	`prod API management administration`
	`prod API management`
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/manage`
	c. Remove the trailing `/manage` from the endpoint:
	`cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod`

Parameters	Description

`--realm provider/identity_provider`	Required.
	The default `identity_provider` is `default-idp-2`.You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):
	
	`apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm`
	
	For example,
	
	`apic identity-providers:list --scope provider --server myserver.com --fields title,realm`
	`total_results: 2`
	`results:`
	`- title: Cloud Manager User Registry`
	`realm: provider/default-idp-2`
	`- title: Corporate LDAP user registry`
	`realm: provider/corporate-ldap`
	
	The `title` value indicates which identity provider to use; copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`
	.

`--username <org_username>`	Required. A user with permissions to modify the provider organization; typically this is the owner or administrator of the organization.
`--password <org_password>`	Required. The organization owner or administrator password.

2. Push the data for the provider organization.
   In the same directory where you ran the `archive:unpack-v10` command, run the following command to push the ported data to your API Manager. See the parameters in Table 2 as well as the usage notes that follow the table.

   Syntax:

   `./apicm archive:push-v10 <api-manager-ui-endpoint> cloud-v10/provider-orgs/<pOrg_name> [optional flags]`

   Example for standalone deployment (optional flags are omitted):

   ```
   ./apicm archive:push-v10 api-manager-ui.mgmt.dev.apic10-stack.example.test cloud-v10/provider-orgs/production_Org
   ```

   Example for Cloud Pak for Integration (optional flags are omitted):

   ```
   ./apicm archive:push-v10 cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod cloud-v10/provider-
   orgs/production_Org
   ```

Table 2. `archive:push-v10` parameters for provider organizations

Parameters	Description

Parameters	Description
<*api-manager-ui-endpoint*>	Required. The API Manager URL endpoint on the Management server for communication with the API Manager user interface. Standalone deployment Example: **api-manager-ui.dev.apic10-stack.example.test** The endpoint was specified when the Management subsystem was installed. To verify your endpoint setting, do one of the following: • View the **apiconnect-apim-ui** definition in the **apiconnect-up.yml** file in the API Connect installation project directory. • In the Cloud Manager UI, select Settings > Endpoints and view API Manager URL. Cloud Pak for Integration Syntax: **cpd-<*APIC_namespace*>.<*your-company.domain.com*>/integration/apis/<*APIC_namespace*>/<*APIC_instance*>** Example: **cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod** where **e2e** is **<*APIC_namespace*>** and **prod** is **<*APIC_instance*>**. Note: How to obtain the server URL on Cloud Pak for Integration Use the Platform Navigator UI to determine the parameter. Note that you cannot get the endpoint by using **oc get routes** because Zen modifies the route internally. a. Locate the Platform UI panel with entries for the deployment. Identify the row that has a **Type** column entry with the description **API management** b. To display the endpoint, hover over the corresponding **prod** entry in the **Name** column for that row. The URL displays at the bottom of the panel. **Name Type** **prod API management administration** **prod API management** **cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod/manage** c. Remove the trailing **/manage** from the endpoint: **cpd-e2e.apps.test1.cp.example.com/integration/apis/e2e/prod**
cloud-v10/provider-orgs/<*pOrg_name*>	Required. The location of your provider org within the **/cloud-v10** file hierarchy you created when you ran unpacked the data archive; for example: **cloud-v10/provider-orgs/production_Org**
--catalogs=<*list_of_names*>	Optional. Push data for only the specified catalogs if you don't want to push the entire provider organization. Use this parameter to specify the names of the catalogs you want to push (to specify multiple catalogs, use a comma-separated list): **./apicm archive:push-v10 <*api-manager-ui-endpoint*> cloud-v10/provider-orgs/<*pOrg_name*> --catalogs=<*catalog_name1,catalog_name2,catalog_name3*>**
--spaces=<*list_of_names*>	Optional. Push data for only the specified spaces if you don't want to push the entire provider organization. The **--catalogs** parameter must also be specified, and exactly one catalog must be specified. Use this parameter to specify the names of the spaces you want to push (to specify multiple spaces, use a comma-separated list). For example: **./apicm archive:push-v10 <*api-manager-ui-endpoint*> cloud-v10/provider-orgs/<*pOrg_name*> --catalogs=<*catalog_name*> --spaces=<*space_name1*>,<*space_name2*>,<*space_name3*>**
--products=<*list_of_name:version_pairs*>	Optional. Push data for only the specified products if you don't want to push the entire provider organization. The **--catalogs** parameter must also be specified, and exactly one catalog must be specified. If spaces are enabled on the catalog, then the **--spaces** parameter must also be specified, and exactly one space must be specified. Use this parameter to specify the names and versions of the products that you want to push (to specify multiple products, use a comma-separated list of name:version pairs as shown in the following examples. Example: catalog with spaces **./apicm archive:push-v10 <*api-manager-ui-endpoint*> cloud-v10/provider-orgs/<*pOrg_name*> --catalogs=<*catalog_name*> --spaces=<*space_name*> --products=<*prod_name1*>:<*prod_version1*>,<*prod_name2*>:<*prod_version2*>,<*prod_name3*>:<*prod_version3*>** Example: catalog without spaces: **./apicm archive:push-v10 <*api-manager-ui-endpoint*> cloud-v10/provider-orgs/<*pOrg_name*> --catalogs=<*catalog_name*> --products=<*prod_name1*>:<*prod_version1*>,<*prod_name2*>:<*prod_version2*>,<*prod_name3*>:<*prod_version3*>**
--dry-run	Optional. Perform a dry run to test the Admin org migration output. This option creates a project directory similar to **cloud-v10**, but titled **.workdir-<*api-manager-ui-endpoint*>** where you can view how the data will be loaded into your new system. The option also displays the objects that will fail due to existing already or missing dependencies. Use this option to view the resolved URLs for where the data will be loaded, without creating resources on the new system. See Migration troubleshooting. **./apicm archive:push-v10 <*api-manager-ui-endpoint*> cloud-v10/provider-orgs/<*pOrg_name*> --dry-run**

Parameters	Description
`--no-drafts`	Optional. Do not push any draft APIs or products (draft objects are objects that have not been published). `./apicm archive:push-v10 <api-manager-ui-endpoint> <path_to_provider_org> --no-drafts` This option can be used with or without the **--catalogs** parameter.
`--drafts-only`	Optional. Only push ported draft APIs and products, and do not push any published APIs or products (draft objects are objects that have not been published). `./apicm archive:push-v10 <api-manager-ui-endpoint> <path_to_provider_org> --drafts-only` This option cannot be used with the **--catalogs** or **--no-drafts** parameters.
`--overwrite`	Optional. Overwrite any existing draft products, draft APIs, published products and their APIs, and OAuth Providers that are already on the deployment. If you set this parameter to false, these artifacts will be left intact.
`--move-subscriptions`	Optional. When this option is used, the subscriptions associated with v5-compatible-API-based products that were ported to API DataPower Gateway-based products and successfully pushed to the API Connect deployment will have their subscriptions moved to the new products, and the original v5-compatible-API-based products will be retired. This option can also be used on subsequent **archive:push-v10** commands to move the subscriptions even after the migrated products are published. Using the **--catalogs**, **--spaces**, and **--products** parameters with the **--move-subscriptions** parameters lets you narrow the set of products to have their subscriptions moved.

Usage notes:

- On Windows: open a command window with Administrator privileges, and omit the "./" from the command.
- Run the **archive:push-v10** command in the same directory where you ran the **archive:unpack-v10** command, and always push the ported data back to the same deployment where you extracted the data.
- The **login** command obtains a token that expires after 8 hours.
  If the token expires while the **archive:push-v10** command is running, the AMU generates errors. In this case, log in again and re-run the **push** command. The token is refreshed every time you log in. For more information, see migration_troubleshooting.html#migration_troubleshooting__token_expire.
- The **archive:push-v10** command only pushes only the following ported objects: OAuth Providers, draft APIs, draft products, configured gateway services, and published products.
- An alternative method of viewing endpoints is to run the **kubectl get ingress** and get the values from the output. For example, see the values in **bold** in the following sample output:

  ```
  r7c221d8a33-apic-portal-director api.portal.dev.apic10-stack.example.test 80, 443 21h
  r7c221d8a33-apic-portal-web portal.dev.apic10-stack.example.test 80, 443 21h
  re314d70920-apiconnect-api platform.mgmt.dev.apic10-stack.example.test 80, 443 22h
  re314d70920-apiconnect-apim-ui apim.mgmt.dev.apic10-stack.example.test 80, 443 22h
  re314d70920-apiconnect-cm-ui cloud.mgmt.dev.apic10-stack.example.test 80, 443 22h
  re314d70920-apiconnect-consumer-api consumer.mgmt.dev.apic10-stack.example.test 80, 443 22h
  rfe867a858b-dynamic-gateway-service service.gw.dev.apic10-stack.example.test 80, 443 21h
  rfe867a858b-dynamic-gateway-service-gw api.gw.dev.apic10-stack.example.test 80, 443 21h
  rfff5043330-analytics-client ac.dev.apic10-stack.example.test 80, 443 21h
  rfff5043330-analytics-ingestion ai.dev.apic10-stack.example.test
  ```

3. Test your newly ported products and APIs to make sure they are working as expected.
   If necessary make changes and push the updates to your API Connect deployment.

Migrating from v10 to v10 on a different form factor

Migrate an API Connect v10 deployment from one form factor to a different form factor.

If you want to move your API Connect deployment to a different platform or network environment, use the form factor migration process.

For brevity, "form factor migration" is referred to as just "migration" in the rest of this topic.

The typical use cases for the migration process are:

- Moving your API Connect deployment from one platform to another, without changing anything else. For example, if you originally deployed on Kubernetes, but want to move to OpenShift.
- Changing the endpoints of your API Connect deployment. For example, the consumer API or portal web endpoint. It is not possible to update your API Connect endpoints after installation. Also, if you want to move your API Connect deployment to a different network, it is likely that the move requires changing the API Connect endpoints.
- Combination of moving to a different platform and changing endpoints.

You do not need to use the migration process for:

- Changing your API Connect deployment profile. See:
  - Changing deployment profiles on VMware.
  - Changing deployment profiles on Kubernetes and OpenShift.
  - Changing deployment profiles on OpenShift top-level CR.
  - Changing the deployment profile on Cloud Pak for Integration.
- Backup, restore, and disaster recovery. See:
  - Backing up, restoring, and disaster recovery.
  - Backing up and restoring on VMware.

  Note:
  You can use the migration save script **save_v10_source_configuration.py** as an extra backup mechanism, but do not use the migration save script as an alternative to the standard backup process.

Key points and limitations of form factor migration:

- The API Connect version on the target system must match the API Connect version on the source system.
- The migration process applies only to the management and portal subsystems. The analytics and gateway subsystems do not require migration steps. The gateway receives its data from the management subsystem, and analytics database backups can be restored directly on the new deployment.

The supported migration paths are shown in the following table:

Table 1. Supported migration paths for API Connect v10

Source deployment platform of API Connect v10	Target deployment platform
Kubernetes	Cloud Pak for Integration..
Kubernetes	OpenShift Container Platform.
Kubernetes	Kubernetes.
VMware	Cloud Pak for Integration.
VMware	OpenShift Container Platform.
VMware	VMware. From V10.0.7.0 and later.
OpenShift Container Platform	Cloud Pak for Integration, keeping the same endpoints.
OpenShift Container Platform	Cloud Pak for Integration, moving to a different cluster, resulting in endpoint changes.
OpenShift Container Platform	OpenShift Container Platform, moving to a different cluster, resulting in endpoint changes.
Cloud Pak for Integration	Cloud Pak for Integration, moving to a different cluster, resulting in endpoint changes.

- **v10 migration overview**
  Use migration scripts to migrate API Connect to another form factor, or to a different deployment on the same form factor.
- **v10 migration requirements**
  Ensure your deployment meets the migrations prerequisites and requirements.
- **v10 migration steps**
  Complete the steps that are required for your v10 migration path.
- **Migrating from OpenShift to Cloud Pak for Integration on the same cluster**
  Migrate your API Connect v10 deployment from OpenShift to Cloud Pak for Integration, while remaining on the same cluster.
- **v10 migration troubleshooting**
  Troubleshoot problems with v10 migration.
- **Shutting down the Postgres database**
  V10.0.6 only. Shutdown the Postgres database to ensure no changes are made to the database during migration.

v10 migration overview

Use migration scripts to migrate API Connect to another form factor, or to a different deployment on the same form factor.

The migration process consists of the following steps:

1. Extract the data from the management and portal subsystems databases of your source system, by using the provided Python script **`save_v10_source_configuration.py`**.
2. Install your target management and portal subsystems, setting new endpoints if required.
3. Specify the mapping of any changed endpoints in a YAML file.
4. Restore the data from your source system, by using the provided Python restore scripts.

Location of v10 migration scripts

You can obtain the scripts from Fix Central, search for **`apiconnect-operator-release-files_<version_number>`**. The scripts are located in the helper_files/formFactorMigration directory:

```
apiconnect-operator-release-files_VERSION_NUMBER/helper_files/formFactorMigration
```

On Kubernetes, OpenShift, and Cloud Pak for Integration, the migration scripts are run from a client computer that has access to the Kubernetes or OpenShift cluster. Scripts are supported on Linux and macOS.

On VMware, the migration scripts are run directly on the API Connect VMs.

v10 migration planning

Consider the following information when you plan your v10 migration:

- **Data migration**
  There are two ways to migrate data:
  - Perform complete migration of data:

    Before starting migration steps on the target system, have the target system ready with the same or larger topology than what is present in the source system. This means the number of Gateways, Portals, and Analytics subsystems must be same or more than what is present in the source system. Start the migration process, map the endpoints between source and target systems, and complete the migration process. All data is migrated. The target system must be validated and tested.

    If you do not want to migrate data for some subsystems on the source system, skip those endpoints in the mapping. Then, after migration is complete, manually delete the data from Cloud Manager and the API Manager UI on the source system.

- Perform migration in more than one step
  In this scenario, at the time of migration the API Connect target system does not have all the subsystems ready when compared to the total number of subsystems in the source system. Migration is run only for the Gateway and Portal subsystems that are available on the target system. At this point, the target system will still have some stale data related to the subsystems that are not migrated. You can then create additional subsystems (Gateways, Portals, or Analytics) on the target system, and run the necessary migration steps to complete the remaining migration process.

  - If additional Gateways or Analytics are added to the source system after completing the initial migration, you must:
    1. Add the new Gateway and Analytics related endpoints in the mapping file.
    2. Register Gateways and Portals on the target system. Complete Step 3 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
    3. Update the gateways in the catalog to point to the new Gateway systems. Complete Step 6 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
  - If additional Portals are added to the source system after completing the initial migration, you must:

    1. Add the new Portal related endpoints in the mapping file.
    2. Register gateways and portals on the target system. Complete Step 3 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
    3. Update the portals in the catalog to point to the new portal systems. Complete Step 4 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
    4. Restore portal backups for each site on the namespace where the new portal is installed. Complete Step 5 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
  - The stale data for which migration is not complete will remain on the target system until you install the additional subsystems and complete migration, or until you use the Cloud Manager or API Manager UI to manually delete it.

- Reuse of Gateways from the source system:
  1. Export the **`ingress-ca`** secret from the source system using the **`-export_cert`** flag in Preparing the source system. The secret is saved to **`data/MANAGEMENT_SUBSYSTEM_NAME/ingress-ca_secret.yaml`**.
  2. Install the **`ingress-ca`** secret manually on the target cluster. This step must be completed before you begin Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
     - **If using top-level CR on OpenShift:** The top-level CR name you are going to use while installing API Connect must be used while creating the secret name and its contents.
     - The name of the ingress-ca secret must be ***`TOP_CR_NAME`*`-ingress-ca`** in case of OpenShift using top-level CR, or CP4I. For Kubernetes or OpenShift using individual subsystems, it must be **`ingress-ca`**.
     - An example of an **`ingress-ca`** secret is given in the following example. If top-level CR is used, replace **`TOP_CR_NAME`** with the top-level CR name you are going to use to install API Connect. For Kubernetes or OpenShift using individual subsystems, all the values will be just **`ingress-ca`**.
     - After modifying the contents of **`ingress-ca`** and saving the yaml, create the secret using **`oc create -f filename.yaml -n NAMESPACE`**.

     ```
     apiVersion: v1
     data:
       ca.crt: CA_CRT
       tls.crt: TLS_CRT
       tls.key: TLS_KEY
     kind: Secret
     metadata:
       annotations:
         cert-manager.io/alt-names: ''
         cert-manager.io/certificate-name: TOP_CR_NAME-ingress-ca
         cert-manager.io/common-name: ingress-ca
         cert-manager.io/ip-sans: ''
         cert-manager.io/issuer-group: ''
         cert-manager.io/issuer-kind: Issuer
         cert-manager.io/issuer-name: TOP_CR_NAME-self-signed
         cert-manager.io/uri-sans: ''
       name: <TOPCR>-ingress-ca
     type: kubernetes.io/tls
     ```

  3. Run the migration steps, starting with Step 1 in Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration. Install API Connect, using the same TOP_CR_NAME (if using top-level CR), by following the remainder of the steps in Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
  4. When you complete Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration, do not map the source system gateways that are to be reused. Just skip them.
  5. Once the migration is complete, restart the gateways pods on the source system, wait for few minutes, and run the health check script to check the status of webhooks.
  6. Validate the APIs running on the source system gateways.
- Delta changes
  Once the migration procedure is started, any delta changes in the source system databases are not handled by the migration scripts. These changes must be handled manually.

  - To avoid delta changes, shut down the Postgres database in the management subsystem. See Shutting down the Postgres database.
    If you shutdown Postgres, you won't be able to login to the any of the API Connect UIs, or Toolkit CLI.

  - If there are delta changes to the management database after saving the configuration for the source system, you can optionally take another backup of the Management subsystem, just prior to restoring the Management subsystem. See Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
- Sizing
  Before installing the API Connect target system, make sure proper capacity planning is done to handle the production workloads. Ensure you have sufficient memory, CPU, and disk size as needed, based on the deployment profile being installed.

- Expected migration time
  The time required to complete migration can depend on the following:

  - The size of the management database. Backup and restore are dependent on this size.
  - Migration of the Portal and Gateway data in all the provider orgs depends on:
    - Number of provider orgs
    - Number of catalogs in these provider orgs

        - Whether Portals are configured for these catalogs
        - Total number of APIs published
    - Creation of Portal sites and restore of Portal backup depends on:
        - Deployment profile (n1, n3 etc) being used
        - Disk performance
        - Total number of Portal sites being create or restored
        - The amount of customization in a Portal site. Customization plays a role in the time needed to restore.

v10 migration requirements

Ensure your deployment meets the migrations prerequisites and requirements.

Subsystem requirements

All subsystems must be in a healthy state.

You must manually create storage classes like block storage or file system storage. Applies to all subsystems and, when installing on CP4I, to Platform Navigator.

Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

- Management subsystem:
    - **v10.0.6 only:** You must configure Management subsystems with either S3 or SFTP backups.
    - **v10.0.6 only:** If s3 storage is being used, postgres in the Management subsystem must be stopped on the source system.
      Note:
      Shutting down postgres is necessary in this scenario because two APIC deployments cannot both use the same S3 backup location. The target API Connect deployment will not start if postgres on the source deployment is up and connecting to the same S3 backup location.

      When postgres is stopped, there is no impact on the published APIs in the gateways. However, the Cloud Manager and API Manager UI will not available because the underlying microservices are down. Users are able to browse the APIs in the Portal. But new consumer orgs or applications or users cannot be created.

    - If the Postgres cluster name on the target is different than the source system, restore will fail. For more information on configuring the target system, see Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
- Developer Portal subsystem:
    - Developer Portal subsystems must be configured with either S3 or SFTP backups.
    - For Developer Portal subsystem, you must have all the mappings between old and new endpoints and hostnames.
- Gateway subsystem:
    - If a custom gateway image is being used on the source system, the same image must be used on the target system
    - For a Gateway subsystem, you must have all the mappings between old and new endpoints and hostnames.

Scripting requirements

- Python 3 and PyYAML python module must be present on the system where the scripts are run.
  To install the PyYAML module:

  ```
  pip3 install pyyaml
  ```

- Do not have any spaces in the complete directory path where the scripts are executed.
- Ensure that the API Connect version of the `apic` toolkit is the same version as for the scripts, and that the toolkit is located in the path where the scripts are executed. Validate toolkit commands to make sure the Cloud Manager and API Manager hostnames are resolved and able to run few commands. Make sure apic toolkit commands are working before trying out the scripts.
- If you are using hostnames, you may need to update the /etc/hosts file for the apic toolkit to work.
- When the scripts are run on the target system, the configuration (data directory) saved from the source system must be available in the location where scripts are present.
- Save the output logs for every script being executed.

Credentials requirements

- In all the scripts where the credentials are passed on the command line, these credentials are the admin org (Cloud Manager) credentials.
- You must be able to use `kubectl` (for native Kubernetes) or `oc` (OpenShift) commands to access the source system cluster and the target system cluster.
- To administer the admin org, you must have credentials with administrative role to access Cloud Manager.
- For provider orgs, if you do not use the migration user, and plan to use provider org credentials, you must have the provider org credentials for the API Manager interface.
- You must manually create:
    - Secrets for entitlement keys to download images from Docker registry.
    - DataPower admin credentials and secret, if the target system is native Kubernetes.
- After migration, the users on the target system will be in the same user registry as in the source system. For example, in CP4I, users from a local user registry (LUR) are not moved to a Zen user registry.

How to obtain hostnames

The hostnames used in the scripts (`--server` or `-s` or `-api_manager_hostname`) and also in the files `gateway_portal_mapping.yaml` or `provider_org_credentials.yaml` can be obtained using the following commands.

The command must be run on the source cluster or target cluster, based on the step for which the hostname is needed.

- Kubernetes:

  ```
  kubectl get ingress -n <NAMESPACE>
  ```

- OpenShift and Cloud Pak for Integration:

  ```
  oc get routes -n <NAMESPACE>
  ```

Example output on Kubernetes:

```
kubectl get ingress -n ask1
NAME                       CLASS    HOSTS                                                ADDRESS
PORTS     AGE
analytics1-ac-endpoint     <none>   ac1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
analytics1-ai-endpoint     <none>   ai1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
apigw1-gateway             <none>   rgw1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d5h
apigw1-gateway-manager     <none>   rgwd1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d5h
management-admin           <none>   admin.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
management-api-manager     <none>   manager.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
management-consumer-api    <none>   consumer.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
management-platform-api    <none>   api.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
portal1-portal-director    <none>   api.portal1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
portal1-portal-web         <none>   portal1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d7h
v5gw1-gateway              <none>   gw1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,1.2.104.161,1.2.104.166   80, 443   5d5h
v5gw1-gateway-manager      <none>   gwd1.testuser1-master.fyre.example.com
1.2.104.133,1.2.104.137,1.2.104.147,1.2.104.157,1.2.104.158,
```

How to obtain realm values

The value of realm (**--realm** or **-r**) used while running the scripts can be obtained as shown in the following examples. The realm values are for Cloud Manager (admin org) and provider org.

- Example for Cloud Manager (admin org):
  The value is passed as a command line option while running the script.

  ```
  apic identity-providers:list --scope admin --fields title,realm --server api.test1-master.fyre.example.com
  total_results: 3
  results:
    - title: Cloud Manager User Registry
      realm: admin/default-idp-1
    - title: googleoidc
      realm: admin/googleoidc
    - title: ldap1
      realm: admin/ldap1
  ```

- Example for provider org realm values:
  The value is used in **provider_org_credentials.yaml** while updating Portal and Gateway information in the provider orgs, when the default migration user is not used. See Step 4 and Step 6 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

  ```
  apic identity-providers:list --scope provider --fields title,realm --server api.testuser1-master.fyre.example.com
  total_results: 3
  results:
    - title: API Manager User Registry
      realm: provider/default-idp-2
    - title: googleoidc
      realm: provider/googleoidc
    - title: ldap1
      realm: provider/ldap1
  ```

Backing up subsystems

It is necessary to backup the management and portal subsystem databases on the source system:

- Backing up and restoring on VMware
- Backing up, restoring, and disaster recovery on Kubernetes and OpenShift

Limitations

1. Analytics service and data is not migrated.
2. If migrating from v10 non-CP4I environment (source apic cluster) to v10 CP4i environment (target apic cluster), make sure the Cloud Manager password for **admin** user is not **7iron-hide** on the source cluster. If default password **7iron-hide** is used, migration will fail after step 3 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration, as configurator job assumes that the install is a fresh installation, and changes the password automatically.

v10 migration steps

Complete the steps that are required for your v10 migration path.

Before you begin, ensure you have reviewed v10 migration overview and v10 migration requirements.
If migrating from OpenShift top-level CR to Cloud Pak for Integration on the same cluster, do not follow the steps here. Instead, refer to Migrating from OpenShift to Cloud Pak for Integration on the same cluster.

- **Preparing the source system**
  Run the **save_v10_source_configuration.py** script to gather the source system configuration.
- **Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration**
  On the target system, configure a cluster, create the necessary secrets, and install API Connect.
- **Installing the target system on VMware**
  Create the project directory, configure you target deployment, create the ISO files, and deploy your VMs.
- **Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration**
  Restore the saved configuration and backups from the source system onto the target system.
- **Restoring the target system on VMware**
  Restore the saved configuration and backups from the source system onto the target system.
- **Verifying the target system on Kubernetes, OpenShift, and Cloud Pak for Integration**
  Verify that the target system is in a healthy state and fully configured.
- **Verifying the target system on VMware**
  Verify that the target system is in a healthy state and fully configured.

Preparing the source system

Run the **save_v10_source_configuration.py** script to gather the source system configuration.

Before you begin

Review v10 migration requirements.

About this task

The **save_v10_source_configuration.py** script:

- Creates a data directory with all configuration saved in it.
- Generates two YAML files in the same directory as the script:
  - **gateway_portal_mapping.yaml**
  - **provider_org_credentials.yaml**

  Important: The YAML files must be filled in:
  - gateway_portal_mapping.yaml must be filled in if, when restoring onto the target system, you choose to run **register_gateway_portals_in_target.py** in silent mode. See step 3.c in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
  - provider_org_credentials.yaml must be filled in if, when restoring onto the target system, you choose not to use the migration user credentials while updating Portal and Gateway information. See step 4 and step 6 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

To view script parameters:

```
python3 save_v10_source_configuration.py --help
```

Procedure

1. Create a directory called **FFM** and copy all python scripts to this directory.
2. Kubernetes, OpenShift, or Cloud Pak for Integration: Save configuration from the source system.
   To save the source system configuration, run **save_v10_source_configuration.py** with the appropriate parameters for your deployment:
   - If all subsystems are in the same namespace, and you are using username/password authentication.
     This command saves configuration of all the installed subsystems in the namespace:

     ```
     python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n NAMESPACE
     ```

   - If a subsystem needs to be skipped so that its configuration is not saved:
     - Skip Portal:

       ```
       python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n
       NAMESPACE -skip_ptl
       ```

     - Skip Analytics:

       ```
       python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n
       NAMESPACE -skip_a7s
       ```

   - If all subsystems are in the same namespace, and you are using SSO authentication:

     ```
     python3 save_v10_source_configuration.py --sso -s PLATFORM_API_HOSTNAME -n NAMESPACE
     ```

   - If all subsystems are in different namespace, and you are using username/password authentication:

     ```
     python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns
     <mgmt_namespace>  -ptl_ns <ptl_namespace> -gw_ns <gw_namespace> -a7s_ns  <a7s_namespace>
     ```

   - If the cluster has multiple Gateways, Portals, or Analytics subsystems distributed across namespaces in the cluster, you can save the configuration of the source system using one of the ways shown below. Multiple namespaces need to be separated by "|" symbol. The namespaces used in below example

command are `ns1`, `ns2`, `ns3`. Both commands work the same way.

- `python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -n "ns1|ns2|ns3"`
- `python3 save_v10_source_configuration.py -u USERNAME -p  PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns "ns1" -ptl_ns  "ns1|ns2" -gw_ns "ns1|ns2|ns3" -a7s_ns "ns1|ns2"`

- If ingress-ca certificates must be saved or exported in the configuration, run the previous commands with `-export_cert` flag. Applicable only when you are saving management subsystem configuration.

3. VMware: Save configuration from the source system.
   On VMware appliance, save the source configuration by running the `save_v10_source_configuration.py` script on the Management and Portal nodes.

   If you have a three replica deployment, run the `save_v10_source_configuration.py` script on just one of the VMs in the cluster.

   a. Copy the `apic` toolkit and `apicup` installer to the management node.
   b. Copy the script `save_v10_source_configuration.py` to the management node:
      - If `ingress-ca` and `root-ca` certificates must be saved in the configuration, copy this script to the project directory for this appliance and upload the project directory to the appliance. The script must be run from inside the project directory.
      - If `ingress-ca` and `root-ca` certificates are not needed, just copy this script to the appliance.
   c. Change to root user (`sudo -i`) and set PATH variable to the directory where the **apic** toolkit is present. This directory must be at the beginning of the path.
   d. Change to the directory where script is located. Check the PATH to verify that the **apic** toolkit is present. To validate, use `apic login` and `apicup version`.
   e. Collect source system configuration. If certificates needed, add the flag `-export_cert` to the command.

   ```
   python3 save_v10_source_configuration.py -u USERNAME -p PASSWORD -s PLATFORM_API_HOSTNAME -r REALM_NAME -mgmt_ns default
   ```

   f. On the Management node, from the location where the script is run, download the data directory, gateway_portal_mapping.yaml, and provider_org_credentials.yaml files. Copy them to the location where the form factor migration scripts are present.
   g. Copy the `save_v10_source_configuration.py` script to the Portal node and run the script as root user, using the following command:

   ```
   python3 save_v10_source_configuration.py -ptl_ns default -skip_mgmt
   ```

   On the Portal node, from the location where the script is run, download the `data` directory and merge it with the `data` directory downloaded from the Management node.
   If multiple Portal services are configured with the Management subsystem, repeat the same operation for each Portal.

What to do next

Configure your target deployment:

- Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration
- Installing the target system on VMware

Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration

On the target system, configure a cluster, create the necessary secrets, and install API Connect.

Before you begin

- Complete Preparing the source system.
- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

About this task

Run the `create_secrets_in_target.py` script to create the necessary secrets. If necessary, shutdown the Postgres database, and then install API Connect, either manually or by running the `install_apic_on_ocp.py` script.

Note: For data migration, you have the option of performing a complete migration at once, or performing migration in more than one step. To review these options, see v10 migration planning.

Procedure

1. Configure a target cluster either Kubernetes or OpenShift or Cloud Pak for Integration, where you intend to migrate the source system data.
   - Note that this configuration is for the cluster only. No API Connect software is installed yet.
   - Make sure `KUBECONFIG` is set to this target cluster, so you can access the cluster using `oc` or `kubectl`.
2. Create the namespaces in the target cluster where you want the API Connect subsystems to run.
   - The namespaces can be different than the source system values.
   - If you have additional subsystems to be installed in different namespaces, create these namespaces also.
3. Run `create_secrets_in_target.py` to create the following secrets:
   - Database backup secrets for Management and Portal subsystems
   - Application credentials
   - Encryption secrets from the source system

   Usage examples:
   - If both Management and Portal are in the same namespace, or if Portal is not installed.

```
python3 create_secrets_in_target.py -n NAMESPACE
```

- If both Management and Portal are in different namespaces:

  ```
  python3 create_secrets_in_target.py -mgmt_ns <MGMT_NAMESPACE> -ptl_ns <PTL_NAMESPACE>
  ```

- If you are creating multiple Portal subsystems in the target API Connect system, to match the configuration of the source system, you must run the script more than once for each Portal subsystem from the source system.
  The script detects that there are two Portal subsystems in the source system, and prompts you to select the subsystem name that matches the target Portal, as shown in the following commands:

  - For Management and first Portal on the target system:

    ```
    python3 create_secrets_in_target.py -mgmt_ns <MGMT_NAMESPACE> -ptl_ns <PTL_NAMESPACE1>
    ```

  - For the second Portal on the target system:

    ```
    python3 create_secrets_in_target.py -skip_mgmt -ptl_ns <PTL_NAMESPACE2>
    ```

4. If applicable to your deployment, shutdown the Postgres database on your source system.
   You can shutdown the Postgres database now, to ensure that there are no delta changes to your source system while the remainder of the migration process takes place.

   Important: V10.0.6.0 only: If s3 backup is being used for the Management subsystem in the source system, the postgres database MUST be shutdown before launching the target API Connect cluster. If it is not shutdown, the launch of the target apic cluster will fail.
   For steps to shutdown Postgres database on Kubernetes or OpenShift, see Shutting down the Postgres database.

5. Launch the target API Connect system.
   a. Review prerequisites:
      - Have a new OpenShift cluster ready with enough resources (CPU, memory, and disk) as required for the installation profile you will use.
      - As needed, create an entitlement key to download the images related to API Connect.
      - The target system must match the topology (number of Gateways, Portals and Analytics) of the source system to ensure that all the data in the Management database is migrated. If all the data in the Management database is not migrated, the not migrated (stale) data must be manually deleted. Alternatively, you can attempt the remaining migration later after installing the additional Portals and Gateways. See v10 migration planning.

   Important: V10.0.6.0 only: If s3 backup is being used for the Management subsystem in the **target** system, once the Management subsystem is ready, you will see that the backup job pods and stanza-create job pods are in `Error` state. This is expected behavior. Once management backup is restored, the error resolves and backups will work. Since the backup will **not** happen until the Management database is restored, it is very important to restore Management subsystem backup on the target system (see step 2 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration) immediately after installing the target system. Any delay can cause the disk to fill up with write-ahead log (WAL) files.
   To review how to monitor Postgres db disk usage, see:

   - Monitoring Postgres disk usage on OpenShift
   - Monitoring Postgres disk usage on VMware
   - Monitoring Postgres disk usage on Kubernetes.

   b. Launch the target API Connect system either manually or by using the `install_apic_on_ocp.py` script:

      - To manually launch from the OpenShift cluster UI:

        This method is the recommended way to launch the target API Connect system for production systems.

        This method is also used for advanced customization or when using custom certificates while launching the target API Connect system.

        - Install and launch the target API Connect cluster. See the instructions for your deployment type:
          - API Connect on OpenShift: Deploying on OpenShift and Cloud Pak for Integration
          - Cloud Pak for Integration: Installation procedures with IBM Cloud Pak for Integration
        - Before creating the top level CR, use the yaml view to add the properties which you saved from the source system configuration.
          For the Management subsystem, the saved configuration is in `data/config.yaml`. For the Portal subsystem, the saved configuration is in `data/config_portal.yaml`.

          Important: If you do not use the configuration from the source system, the migration fails.
          - Management subsystem properties:
            - encryptionSecret
            - Management subsystem name
            - site name
            - database backup credentials
            - custom application credentials
            - originalUID
          - Portal subsystem properties:
            - encryptionSecret
            - site name
            - database backup credentials
            - originalUID

          Note: If target system is Kubernetes, the properties must be in the Management and Portal subsystem CRs.
        - The top level CR must look like the following example. The values for this top CR are from the generated artifacts (data/config.yaml and data/config_portal.yaml) after running Preparing the source system. Note that the example shows a value for `spec.management.name`. You can obtain this value from config.yaml.

          ```
          apiVersion: apiconnect.ibm.com/v1beta1
          kind: APIConnectCluster
          metadata:
            labels:
              app.kubernetes.io/instance: apiconnect
              app.kubernetes.io/managed-by: ibm-apiconnect
              app.kubernetes.io/name: apiconnect-minimum
            name: c1
            namespace: test1
          ```

```
spec:
  analytics:
    storage:
      enabled: true
      type: unique
  license:
    accept: true
    license: SOME_KEY
    use: nonproduction
  management:
    customApplicationCredentials:
    - name: atm-cred
      secretName: management-atm-cred
    - name: ccli-cred
      secretName: management-ccli-cred
    - name: cui-cred
      secretName: management-cui-cred
    - name: dsgr-cred
      secretName: management-dsgr-cred
    - name: juhu-cred
      secretName: management-juhu-cred
    - name: cli-cred
      secretName: management-cli-cred
    - name: ui-cred
      secretName: management-ui-cred
    databaseBackup:
      credentials: ftp-mgmt-backup-secret
      host: SFTP_SERVER_HOST
      path: /root/backup/v10_management
      port: 22
      protocol: sftp
      restartDB:
        accept: false
    encryptionSecret:
      secretName: management-enc-key
    name: management
    originalUID: d435b534-ce93-463a-9005-692c2af94df7
    siteName: be023200
  portal:
    encryptionSecret:
      secretName: portal1-enc-key
    originalUID: 1adc672a-0e77-4890-b476-c41fcedb0b64
    portalBackup:
      credentials: ftp-mgmt-backup-secret
      host: 1.2.3.4
      path: /root/backup/portal1
      port: 22
      protocol: sftp
    siteName: cb06099f
  profile: n1xc7.m48
  storageClassName: rook-ceph-block
  version: 10.0.6.0
```

    - You can also use the script **`install_apic_on_ocp.py`** with **`-no_install`** flag to generate the top level CR yaml, and use the yaml in the OpenShift UI and customize further.
- To launch using the **`install_apic_on_ocp.py`** script:
  The script is used to launch API Connect on OpenShift or API Connect with Cloud Pak for Integration using a top level CR that specifies one of each subsystem type:
    - Management
    - Developer Portal
    - Analytics
    - API Gateway

  Prerequisite:
    - The OpenShift cluster must be ready, and accessible using **`oc`**.
    - **Important:** The script is used to launch **ONLY** the specific version of API Connect based on the downloaded version of the artifacts.

  Usage:

    - Run the script with **`--help`** option to show additional flags available like **`-production and -profile`**.
    - License use is set to **`nonproduction`** by default unless the flag **`-production`** is used while running this script.
    - The profile defaults to the one replica profile **`n1xc7.m48`**, to specify a different profile use the **`-profile`** flag. For details of the available profiles, see API Connect deployment profiles for OpenShift and Cloud Pak for Integration. The one replica profile deploys 1 replica of each pod and is only for light, non-critical workloads such as basic development and testing.
    - Optionally, you can use the **`-no_install`** flag if you want to generate the yaml files (top level CR and other files needed in installation) without installing. In the following example commands, add the **`-no_install`** flag to the existing command parameters. You can use this top level CR from the OpenShift UI, and create an API Connect cluster.
    - Note : If CP4I environment, check if configurator job is completed and also check the output of:

      `oc get apiconnectcluster -n NAMESPACE`

Example command invocations:

Note: Each of the example commands includes parameters for a three replica profile deployment: **`-production -profile n3xc16.m48`**. If you have a one replica profile, omit these parameters.

- API Connect on OpenShift:
    - If the apiconnect and DataPower operators must be installed in all namespaces:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME -production -profile n3xc16.m48
      ```

    - If the apiconnect and DataPower operators must be installed in a specific namespace:

```
python3 install_apic_on_ocp.py -license LICENSE_VALUE -n NAMESPACE  -name topCRName -storageclass_apic
STORAGE_CLASS_NAME  -operator_in_specific_namespace -production -profile n3xc16.m48
```

- API Connect in Cloud Pak for Integration:
  - If operators are in all namespaces:
    - V10.0.7.0 and later:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -production -profile n3xc16.m48
      ```

    - V10.0.6.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -production
      -profile n3xc16.m48
      ```

  - If operators are in a specific namespace:
    - V10.0.7.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -operator_in_specific_namespace -production -profile n3xc16.m48
      ```

    - V10.0.6.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -
      operator_in_specific_namespace -production -profile n3xc16.m48
      ```

    - V10.0.7.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -operator_in_specific_namespace -production -profile n3xc16.m48
      ```

    - V10.0.6.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -
      operator_in_specific_namespace -production -profile n3xc16.m48
      ```

  - If operators are in all namespaces, with additional flags **`-production`** (for license to be set to production) and **`-profile`** (using n3 profile in this example):
    - V10.0.7.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -production -profile n3xc16.m48
      ```

    - V10.0.6.0:

      ```
      python3 install_apic_on_ocp.py -license LICENSE_VALUE1 -n NAMESPACE -name topCRName -storageclass_apic
      STORAGE_CLASS_NAME1 -cp4i -storageclass_pn STORAGE_CLASS_NAME2 -license_pn LICENSE_VALUE2 -production
      -profile n3xc16.m48
      ```

6. Optional: Install any additional Gateways, Developer Portals, or Analytics subsystems that are needed.
   - Complete this step now if your target system will have additional Gateways, Developer Portals, or Analytics subsystems to match what is there in the source API Connect system, and you want to complete data migration for all subsystems in one pass through the procedures in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
   - Skip this step now, and proceed to step 7 if either of the following cases are true:
     - If your target system will not need any additional Gateways, Developer Portals, or Analytics subsystems.
     - Your target system will have additional Gateways, Developer Portals, or Analytics subsystems but you want to first complete data migration based on the top CR installation (1 Management, 1 Portal, 1 Gateway, 1 Analytics), before you install any additional subsystems.

   Note: To review the data migration choices, see v10 migration planning.
   If you choose complete this step now, see:

   - Deploying on OpenShift and Cloud Pak for Integration
   - Installation procedures with IBM Cloud Pak for Integration
7. Next, restore the configuration from the source system. Continue with Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

Installing the target system on VMware

Create the project directory, configure you target deployment, create the ISO files, and deploy your VMs.

Form factor migration to VMware is supported from v10.0.7 only, and is only supported from a VMware environment. You cannot migrate a Kubernetes or OpenShift API Connect deployment to VMware.

When you install the target system, before you create the ISO files that are used in the OVA deployment, you must set some of the subsystem properties of your target system to the same values as your source system. Refer to the data directory that you created in Preparing the source system to find these values.

Other subsystem properties can be set to different values from your source subsystem. For example, the API endpoints such as **`portal-www`** and **`platform-api`**. Changing API endpoints is the typical use-case of a VMware to VMware migration.

Management subsystem

Follow the usual steps for configuring your management subsystem deployment: Configuring the Management subsystem, but set the following properties to the same values as your source system.

1. Subsystem name.

```
apicup subsys create <management subsystem name> management
```

where *<management subsystem name>* is the same as your source system.

2. Encryption secret.

```
apicup certs set <management subsystem name> encryption-secret --encoded-string <v10 source mgmt encryption secret>
```

where *<v10 source mgmt encryption secret>* is found in data/*<management subsystem name>*/encryption-secret-*.yaml

3. Site name.

```
apicup certs set <management subsystem name> site-name <v10 source mgmt site name>
```

where *<v10 source mgmt site name>* is found in data/*<management subsystem name>*/config.yaml

4. Credential secrets.

```
apicup certs set <management subsystem name> atm-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> consumer-toolkit-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> designer-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> toolkit-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> juhu-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> consumer-ui-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> ui-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> discovery-credential --encoded-string '<encoded string>'
apicup certs set <management subsystem name> governance-credential --encoded-string '<encoded string>'
```

where *<encoded string>* is found in the corresponding secret YAML file in data/*<management subsystem name>*. For example:

```
<management subsystem name>-atm-cred.yaml
<management subsystem name>-ccli-cred.yaml
<management subsystem name>-cli-cred.yaml
<management subsystem name>-cui-cred.yaml
<management subsystem name>-dsgr-cred.yaml
<management subsystem name>-juhu-cred.yaml
<management subsystem name>-ui-cred.yaml
```

The encoded string is stored in `credential.json` property of each file:

```
apiVersion: v1
data:
  credential.json:
eyAiaWQiOiAiMzhmMzRjOTktODFhNC00NWFkLWFkOTgtMGM1MGFjN2YzYmJjIiwgInNlY3JldCI6ICI3OTIxYzE5Yi1jNWUwLTRhYzQtYTkyZC1iZjNmYWJlZWEwNDIiIH0=
kind: Secret
...
```

5. You can use the same management database backup configuration as your source system, or you can specify a new configuration.

Portal subsystem

Follow the usual steps for configuring your portal subsystem deployment: Configuring the Developer Portal subsystem, but set the following properties to the same values as your source system.

1. Subsystem name.

```
apicup subsys create <portal subsystem name> portal
```

where *<portalsubsystem name>* is the same as your source system.

2. Encryption secret.

```
apicup certs set <portal subsystem name> encryption-secret --encoded-string <v10 source ptl encryption secret>
```

where *<v10 source ptl encryption secret>* is found in data/*<ptl subsystem name>*/portal-encryption-secret-*.yaml

3. Site name.

```
apicup certs set <portal subsystem name> site-name <v10 source ptl site name>
```

where *<v10 source ptl site name>* is found in data/*<ptl subsystem name>*/config.yaml

4. Backup configuration.
Use the same backup configuration as your source system to restore your portal sites.

Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration

Restore the saved configuration and backups from the source system onto the target system.

Before you begin

- Complete Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
- Ensure that any network route between the target installation and the source system is disabled. The target system must not be able to communicate with the source system.

About this task

Use scripts to:

- Restore the Management subsystem
- Register Gateways and Portals on the target system
- Update Portals in the Catalog to point to new Portal systems
- Restore Portal backups for each site
- Update Gateways in the Catalog to point to new Gateway systems

All scripts run on the target system must be run from a location where the data directory is present so that the source configuration can be used.

Note: You can optionally run scripts in `--silent` mode, particularly for the following scenarios:

- Register Gateways and Portals in the target system. (Step 3)
- Update Portals in the Catalog to point to new Portal systems. (Step 4)
- Update Gateways in the Catalog to point to new Gateway systems. (Step 6)

If there are multiple Portals and Management subsystems present in the data directory (as saved during Preparing the source system), the restore script prompts you to select the subystem name to use for migration while running the script. If running in silent mode, that value must be supplied as an additional flag.

Deployments with multiple Portals and Gateways are a common scenario. The saved data can have multiple Management subsystems only if you ran **`save_v10_source_configuration.py`** against multiple clusters with different Management subsystem names, without cleaning the data directory.

Procedure

1. Before restoring the Management subsystem backup, determine whether there have been any delta changes to your source system since you backed up the Management subsystem database and Portal database.
   - If you shutdown the Postgres database, as instructed in Installing the target system on Kubernetes, OpenShift, and Cloud Pak for Integration, there are no delta changes. Continue to step 2.
   - If you did not shutdown the Postgres database, there could be some delta changes to the Management database, because the Management subsystem on the source system remains in READY state. In this case, to ensure you capture the latest configuration, repeat Preparing the source system. By doing so, the Management and Portal databases are backed up again. The latest backup will then be used to restore to the target system.
2. Restore a Management subsystem backup onto the target system.
   a. On the target API Connect system, ensure that the Management subsystem is up and running,
   b. Run **`restore_management_db.py`**. Examples:
      - OpenShift or Kubernetes target system:

        ```
        python3 restore_management_db.py -n <mgmt namespace>
        ```

        or:

        ```
        python3 restore_management_db.py -mgmt_ns <mgmt namespace>
        ```

      - Cloud Pak for Integration target system:

        ```
        python3 restore_management_db.py -n <namespace> -s <platform API hostname>
        ```

      - If for some reason the management subsystem is not healthy, restore using the command:

        ```
        python3 restore_management_db.py -n <mgmt namespace> -ignore_health_check
        ```

3. Register Gateways and Portals on the target system.
   a. Prerequisites:
      - The **`register_gateway_portals_in_target.py`** script must be run on the target system after the Management database from the source system has been restored.
      - The Gateway and Developer Portal subsystems that need to be registered must be in a healthy state.
   b. Usage notes:
      - The credentials provided from this step will be same as the credentials used on the source system, because the Management database from the source system is restored.
      - Keep the endpoints ready for the new Gateway and Portal services so that they can be registered by running **`register_gateway_portals_in_target.py`**. To view endpoints, use **`oc get routes`**.
      - If you run **`register_gateway_portals_in_target.py`** for a Cloud Pak for Integration target system, and you see the error **`The API Connect Gateway Service is already registered to another API Manager instance`**, restart all the gateway pods, wait for them to become healthy, and run **`register_gateway_portals_in_target.py`** again.
      - Make sure the mapping of old URLs with the new ones are correct. For example, when the source system is a VMware appliance, if wrongly mapped (wrong values provided) communication errors occur during Gateway registration. Example errors messages:

        - **`"Error: An error occurred communicating with the gateways subsystem "`**
        - **`"error: 'Client network socket disconnected before secure TLS connection was established'"`**

        On the appliance, the Gateway Manager URL uses port **`3000`**.

   c. Run **`register_gateway_portals_in_target.py`** in either interactive mode or silent mode:

      - Interactive mode
        The **`register_gateway_portals_in_target.py`** script displays the Portal and Gateway endpoints from the source system, and prompts the user to enter the corresponding values from the target system. Examples:

        - If Management, Portal and Gateway subsystems are in the same namespace:

```
python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r
<realm name> -n <namespace>
```

- If Management, Portal and Gateway subsystems are in different namespaces, use `-mgmt_ns`, `-ptl_ns`, and `-gw_ns` to specify each, for example:

  ```
  python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r
  <realm name> -mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -gw_ns <gway namespace>
  ```

- If you are using SSO authentication, replace:

  ```
  -u <username> -p <password>
  ```

  with:

  ```
  --sso
  ```

- Multiple namespaces can alternatively be specified with `"ns1|ns2|ns3"` notation. For example:

  ```
  python3 register_gateway_portals_in_target.py -n "ns1|ns2|ns3" -u admin -p <password> -s <platform API
  hostname> -r admin/default-idp-1 -silent
  ```

  ```
  python3 register_gateway_portals_in_target.py -s <platform API hostname> -u admin -p <password> -r
  admin/default-idp-1 -mgmt_ns ns1 -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -silent
  ```

- Silent mode
  The gateway_portal_mapping.yaml file, with mapping between old and new endpoints for Gateway, Portal, and Analytics endpoints, must be located in the same directory as **register_gateway_portals_in_target.py**.

  This gateway_portal_mapping.yaml file is generated when saving the source API Connect system configuration with properties data filled in. For every Gateway, Portal, and Analytics subsystem registered with Management in the source system, the yaml has the endpoint mapping between source and target systems.

  Remaining data must be filled to use the yaml in silent mode. All values must begin with **https://**.

  - Example:

    ```
    python3 register_gateway_portals_in_target.py -n <namespace> -u <username> -p <password> -s <platform API
    hostname> -r <realm name> -silent
    ```

  - Example: If the subsystems are in multiple namespaces.
    For example, ns1, ns2, ns3 namespaces contain the subsystems. Both of the following commands work the same way:

    ```
    python3 register_gateway_portals_in_target.py -n "ns1|ns2|ns3" -u <username> -p <password> -s <platform API
    hostname> -r <realm name> -silent
    ```

    ```
    python3 register_gateway_portals_in_target.py -s <platform API hostname> -u <username> -p <password> -r
    <realm name> -mgmt_ns ns1 -ptl_ns "ns1|ns2" -gw_ns "ns1|ns2|ns3" -silent
    ```

  - Example gateway_portal_mapping.yaml file with:
    - Two Gateways
    - One Portal or Analytics subsystem
    - All data mapped to the target API Connect system:

    ```
    analytics_mapping:
      registered_analytics_name:
        https://ac.source-apic.example.com: https://a1-a7s-ac-endpoint-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
    gateway_mapping:
      registered_apigw_name:
        https://rgw.source-apic.fyre.example.com: https://a1-gw-gateway-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
        https://rgwd.source-apic.fyre.example.com: https://a1-gw-gateway-manager-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
      registered_v5gw_name:
        https://gw.source-apic.fyre.example.com: https://v5gw.gw.apps.vlxocp-3278.cp.fyre.example.com
        https://gwd.source-apic.fyre.example.com: https://v5gw.gwd.apps.vlxocp-3278.cp.fyre.example.com
    portal_mapping:
      registered_portal_name:
        https://api.portal.source-apic.fyre.example.com: https://a1-ptl-portal-director-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
        https://portal.source-apic.fyre.example.com: https://a1-ptl-portal-web-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
    ```

  - Example gateway_portal_mapping.yaml file with:
    - Only two Gateways
    - One Portal or Analytics
    - One Gateway not mapped
      Note that "not mapped" means that the data in the Management subsystem related to this Gateway is not mapped, and the Gateway will not be migrated during this migration phase.

      The Gateway could be migrated later; for example, after the second Gateway is created on the target system. Until the data related to the not-migrated Gateway or Portal is migrated or deleted, there will be stale data in the database.

      The same behavior can be achieved by leaving the values empty, or removing the entire section (name and mapping).

    ```
    analytics_mapping:
      registered_analytics_name:
        https://ac.source-apic.example.com: https://a1-a7s-ac-endpoint-apic1.apps.vlxocp-
    3278.cp.fyre.example.com
    gateway_mapping:
      registered_apigw_name:
    ```

```
    https://rgw.source-apic.fyre.example.com: https://a1-gw-gateway-apic1.apps.vlxocp-
3278.cp.fyre.example.com
    https://rgwd.source-apic.fyre.example.com: https://a1-gw-gateway-manager-apic1.apps.vlxocp-
3278.cp.fyre.example.com
  registered_v5gw_name:
    https://gw.source-apic.fyre.example.com: NEW_GATEWAY_API_ENDPOINT_BASE_IN_TARGET_SYSTEM
    https://gwd.source-apic.fyre.example.com: NEW_GATEWAY_ENDPOINT_IN_TARGET_SYSTEM
portal_mapping:
  registered_portal_name:
    https://api.portal.source-apic.fyre.example.com: https://a1-ptl-portal-director-apic1.apps.vlxocp-
3278.cp.fyre.example.com
    https://portal.source-apic.fyre.example.com: https://a1-ptl-portal-web-apic1.apps.vlxocp-
3278.cp.fyre.example.com
```

4. Use `update_to_new_portals.py` to update Portals in the catalog to point to new Portal systems.
   Note: If there is no Portal subsystem on the source system, skip this step. Also, skip this step if no Portal sites are created in any of the provider orgs. Continue with 6.
   a. Review prerequisites:
      - The Management and Portal subsystems on the target API Connect system must be in a healthy state.
      - The `update_to_new_portals.py` script must be run on the target system after the following actions have been completed:
        - The Management database from the source system has been restored.
        - The new Gateway or Portal instance is registered in the Cloud Manager, using script or from the UI.
      - The credentials for accessing the admin org (Cloud Manager UI) and, if required, the provider org (API Manager UI) must be available for running the script.
   b. Run the `update_to_new_portals.py` script in one of two modes, based on how the Provide Org credentials are available. The modes are:
      - Using a common migration user created by the script
      - Using credentials provided by the user in a yaml file

      Important: The provider org credentials must have administrative access.

   Examples:

   - Using a common migration user created by the script :
     The `update_to_new_portals.py` script runs in this mode by default. This script creates a migration user `muser1` in `migrationur` local user registry, and uses it to migrate Portal information in all the provider orgs. Once the migration is done, the user and user registry are deleted. There is no change in data once the migration is complete.

     Examples:

     - If the Management and Portal subsystems are in the same namespace

       ```
       python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -n
       <namespace> -api_manager_hostname <platform API hostname>  -silent
       ```

     - If using SSO authentication:

       ```
       python3 update_to_new_portals.py -s <platform API hostname> --sso -n <namespace> -api_manager_hostname
       API_MANAGER_HOSTNAME -silent
       ```

     - If the Management and Portal subsystems are in different namespaces:

       ```
       python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns
       <mgmt namespace> -ptl_ns <portal namespace>E -api_manager_hostname <platform API hostname>E -silent
       ```

   - Using credentials provided by the user in a `provider_org_credentials.yaml` file:
     The `provider_org_credentials.yaml` file, with some data filled in, is generated by the `save_v10_source_configuration.py` script you ran in Preparing the source system. The yaml file is available in the same directory as the script. The script has to be edited and all details filled in before it can be used.

     Examples:

     - If Management and Portal subsystems are in the same namespace:

       ```
       python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name>  -n
       <namespace> -no_migration_user -silent
       ```

     - If using SSO authentication:

       ```
       python3 update_to_new_portals.py -s <platform API hostname> --sso -n <namespace> -no_migration_user -silent
       ```

     - If Management and Portal subsystems are in different namespaces:

       ```
       python3 update_to_new_portals.py -s <platform API hostname> -u <username> -p <password> -r <realm name>  -
       mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -no_migration_user -silent
       ```

     Example provider_org_credentials.yaml:

     - With credentials given for every provider org. In this example, only two orgs are present.

       ```
       provider_org_credentials:
         apiManagerHostName: <platform API hostname>
         PORG_NAME_1:
           password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
           realm: provider/default-idp-2
           username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
         PORG_NAME_2:
           password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
           realm: provider/default-idp-2
           username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
         useSameCredentialsForAllProviderOrgs: false
       ```

     - With credentials given for just one provider org. Also requires `useSameCredentialsForAllProviderOrgs` set to true. This method can be used only when the same credentials are used for all provider orgs.

Note: In some cases, there are multiple provider orgs in the Management database.

```
provider_org_credentials:
  apiManagerHostName: <platform API hostname>
  PORG_NAME_1:
    password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
    realm: provider/default-idp-2
    username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
  useSameCredentialsForAllProviderOrgs: true
```

- With SSO credentials and just one provider org:

```
provider_org_credentials:
  apiManagerHostName: <platform API hostname>
  useSameCredentialsForAllProviderOrgs: false
  porg_anme:
    #sso authentication
    #the key(that can expire in sometime) has to be generated separately using
https://<apiManagerHostname>/auth/manager/sign-in/?from=TOOLKIT
    apiKey: some_key
```

5. Restore Portal backups for each site.
   Note: If there is no Portal subsystem on the source system, skip this step. Continue with 6.
   To restore the Portal sites on the target system, you can either use the using script restore_portal_db.py or complete a manual restore.

   - Restoring through the script:

     ```
     python3 restore_portal_db.py -n PORTAL_SUBSYSTEM_NAMESPACE
     ```

   - Restoring through manual steps:
     - Log in to the Portal admin container:

       ```
       oc exec -it <www_pod> -c admin -- bash
       ```

     - List the site backups:

       ```
       remote_transfer_backup -l
       ```

       The list of backups that need to be restored is available in the output logs as a report of Step 4.

     - Download each site backup, and run it for all sites:

       ```
       remote_transfer_backup -d <SITE_BACKUP_FILENAME>
       ```

     - Restore each of the site backups:

       ```
       restore_site -f -u <SITE_URL> <SITE_BACKUP_FILENAME>
       ```

   Note: If there are more than one Portal subsystems on the target system, as needed to correspond to the Portal subsystems on the source system, run the script for each of the matching source and target Portal subsystems.
   For example, if there are two Portals on the target system, matching what is on the source system, and these Portals are installed in namespaces **target1** (for the top-level CR) and **ptl2** (for second portal), then the script will be run twice, as follows:

   ```
   python3 restore_portal_db.py -n target1

   python3 restore_portal_db.py -n ptl2
   ```

6. Run **update_to_new_gateways.py** to update Gateways in the Catalog to point to new Gateway systems.
   a. Review prerequisites:
      - The Gateway subsystem must be in a healthy state.
      - The **update_to_new_gateways.py** script must be run on the target system after the following actions have been completed:
        - The Management database from the source system has been restored.
        - The new Gateway instance is registered in the Cloud Manager, using script or from the UI.
      - The credentials for accessing the admin org (Cloud Manager UI) and, if required, the provider org (API Manager UI) must be available for running the script.

   b. Run the **update_to_new_gateways.py** script in one of two modes, based on how the Provider Org credentials are available. The modes are:
      - Using a common migration user created by the script
      - Using credentials provided by the user in a yaml file

      Important: The provider org credentials must have administrative access.
      - Using a common migration user created by the script:
        The **update_to_new_gateways.py** script runs in this mode by default. This script creates a migration user **muser1** in **migrationur** local user registry, and uses it to migrate Portal information in all the provider orgs. Once the migration is done, the user and user registry are deleted. There is no change in data once the migration is complete.

        Examples:

        - If the Management and Gateway subsystems are in the same namespace:

          ```
          python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name>  -
          n <namespace> -api_manager_hostname <platform API hostname>  -silent
          ```

        - If using SSO authentication:

          ```
          python3 update_to_new_gateways.py -s <platform API hostname> --sso -n <namespace> -api_manager_hostname
          <platform API hostname> -silent
          ```

        - If the Management and Gateway subsystems are in different namespaces:

          ```
          python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name>  -
          mgmt_ns <mgmt namespace> -ptl_ns <portal namespace> -api_manager_hostname <platform API hostname> -silent
          ```

- Using credentials provided by the user in a `provider_org_credentials.yaml` file:
  The `provider_org_credentials.yaml` file, with some data filled in, is generated by the `save_v10_source_configuration.py` script you ran in Preparing the source system. The yaml file is available in the same directory as the script. The script has to be edited and all details filled in before it can be used.

  - If the Management and Gateway subsystems are in the same namespace:

    ```
    python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name>  -n <namespace> -no_migration_user -silent
    ```

  - If using SSO authentication:

    ```
    python3 update_to_new_gateways.py -s <platform API hostname> --sso -n <namespace> -no_migration_user -silent
    ```

  - If the Management and Gateway subsystems are in different namespaces:
    Note that if multiple gateway subsystems present across namespaces, you can give multiple namespaces separated by the "|" symbol for `-gw_ns`.

    ```
    python3 update_to_new_gateways.py -s <platform API hostname> -u <username> -p <password> -r <realm name> -mgmt_ns <mgmt namespace> -gw_ns GW_NAMESPACE -no_migration_user -silent
    ```

  - Example provider_org_credentials.yaml
    - With credentials given for every provider org, and only two orgs present:

      ```
      provider_org_credentials:
        apiManagerHostName: <platform API hostname>
        PORG_NAME_1:
          password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
          realm: provider/default-idp-2
          username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
        PORG_NAME_2:
          password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
          realm: provider/default-idp-2
          username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
        useSameCredentialsForAllProviderOrgs: false
      ```

    - With credentials given for just one provider org, and `useSameCredentialsForAllProviderOrgs` is set to `true`.
      This can be used only when same credentials are used for all provider orgs.

      Note: There can be multiple provider orgs in the Management database.

      ```
      provider_org_credentials:
        apiManagerHostName: <platform API hostname>
        PORG_NAME_1:
          password: PASSWORD_FOR_THIS_REALM_TO_BE_CHANGED
          realm: provider/default-idp-2
          username: USERNAME_FOR_THIS_REALM_TO_BE_CHANGED
        useSameCredentialsForAllProviderOrgs: true
      ```

    - With SSO credentials and just one provider org:

      ```
      provider_org_credentials:
        apiManagerHostName: <platform API hostname>
        useSameCredentialsForAllProviderOrgs: false
        porg_anme:
          #sso authentication
          #the key(that can expire in sometime) has to be generated separately using
      https://<apiManagerHostname>/auth/manager/sign-in/?from=TOOLKIT
          apiKey: some_key
      ```

7. Continue to Verifying the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

Restoring the target system on VMware

Restore the saved configuration and backups from the source system onto the target system.

Restoring the management subsystem

1. Copy the toolkit CLI apic executable to your FFM directory. See Installing the toolkit. Ensure that your toolkit CLI version matches the API Connect version of your target management VM.
   You created the FFM directory in Preparing the source system. The FFM directory contains Python scripts, YAML files, and a data directory that contains your exported source system data.

2. Login to your target management VM with SSH, and switch to the root user:

   ```
   ssh apicadm@<management hostname>
   ```

   ```
   sudo -i
   ```

   If you deployed a three replica management profile, then login to just one of the VMs.

3. Copy the FFM directory to your target management VM.
4. Ensure that the FFM directory is set first in your `PATH` variable:

   ```
   export PATH=<FFM directory>:$PATH
   ```

Note: The API Connect VMs have an executable that is called apic in the **PATH** variable. The toolkit CLI apic must come before it in the **PATH** so that the restore scripts run correctly.

5. Run the management database restore script:

   ```
   python3 restore_management_db.py -n default
   ```

   Tip: To get help information with the restore scripts, run them with the flag **--help**.

   ```
   python3 <script name> --help
   ```

   The script output shows the progress and when the restore is complete.

6. Edit the gateway_portal_mapping.yaml file and set the endpoints on right side of the file to the endpoints defined for your target VM. The endpoint values can be obtained from the apiconnect-up-v10.yml file in your project directory.
7. Run the register gateway and portals script:

   ```
   python3 register_gateway_portals_in_target.py -s <platform API endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -silent
   ```

   where
   - *<platform API endpoint>* is defined in the apiconnect-up-v10.yml file in your project directory.
   - *<password>* is the admin password that is used on your source system.

8. Run the update portals script:

   ```
   python3 update_to_new_portals.py -s <platform API endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -api_manager_hostname <platform API endpoint> -silent
   ```

9. Run the update gateways script:

   ```
   python3 update_to_new_gateways.py -s <platform API endpoint> -u admin -p <password> -r admin/default-idp-1 -n default -api_manager_hostname <platform API endpoint> -silent
   ```

Restoring the portal subsystem

1. Login to your target portal VM with SSH, and switch to the root user:

   ```
   ssh apicadm@<portal hostname>
   ```

   ```
   sudo -i
   ```

   If you deployed a three replica portal profile, then login to just one of the VMs.

2. Copy the FFM directory from the management VM to your portal VM.
3. Ensure that the FFM directory is set first in your **PATH** variable:

   ```
   export PATH=<FFM directory>:$PATH
   ```

   Note: The API Connect VMs have an executable that is called apic in the **PATH** variable. The toolkit CLI apic must come before it in the **PATH** so that the restore scripts run correctly.

4. Run the restore portal sites script:

   ```
   python3 restore_portal_db.py -n default
   ```

Verifying the target system on Kubernetes, OpenShift, and Cloud Pak for Integration

Verify that the target system is in a healthy state and fully configured.

Before you begin

Complete Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

About this task

Check the health of the target system, complete any optional configuration, and validate data, APIs, and URLs.

Procedure

1. Use the **health_check_post_migration.py** script to health check the target system.
   Examples:

   - When all the subsystems are in the same namespace, and the deployments uses username/password authentication:

     ```
     python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -n NAMESPACE
     ```

   - When all the subsystems are in the same namespace, and the deployments uses username/password authentication, with use of the optional flag **-check_portal_site_health** to check the health of Portal sites:

     ```
     python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD  -n NAMESPACE -check_portal_site_health
     ```

   - When subsystems are present in different namespaces, and the deployment uses username/password authentication

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD  -mgmt_ns
NAMESPACE -gw_ns NAMESPACE
```

- When all subsystems are in different namespaces, and the deployment uses username/password authentication:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD  -mgmt_ns
NAMESPACE -gw_ns NAMESPACE -ptl_s NAMESPACE -a7s_ns NAMESPACE
```

- When all subsystems are in the same namespace, and the deployment uses SSO authentication:

```
python3 health_check_post_migration.py -n NAMESPACE --sso -s PLATFORM_API_HOSTNAME
```

- When all subsystems are in the same namespace, and the deployment uses SSO authentication silent mode:

```
python3 health_check_post_migration.py -n NAMESPACE --sso -s PLATFORM_API_HOSTNAME -api_key API_KEY_VALUE
```

- When there are multiple subsystems across namespaces:

```
python3 health_check_post_migration.py -s PLATFORM_API_HOSTNAME -r REALM_NAME -u USERNAME -p PASSWORD -n
"namespace1|namespace2|namespace3"
```

2. Complete any optional configuration steps needed for your deployment:
   a. If necessary, update redirect URLS in the authorization server.
      If OIDC or SSO authentication is being used, the redirect URLS must be updated in the authorization server after migration. For example, for Google OIDC, you must to add the new redirect URLs in Google Developer Console. The new redirect URLs can be found in the user registry in Cloud Manager.
   b. If necessary, complete configuration for Analytics.
      If the Analytics service is present on the target system, backups must be configured. Also, use the Cloud Manager UI to manually associate the Analytics with the Gateways, as needed.
   c. If the target system is Cloud Pak for Integration, you must use the password from the target API Connect system to log in to Platform Navigator.
   d. If necessary, update Portal and Gateway services names in scripts or utilities.
      The migrated Portal and Gateway services will have new names (old_name + "new"). Any scripts and utilities written using the apic toolkit must be updated if they are using names as input parameter and not the ids. This is applicable to only Portal and Gateway names.
3. Validate configuration data:
   a. Validate the data in the target API Connect system by logging into Cloud Manager and API Manager UIs.
   b. Validate the APIs published in the Gateways.
   c. Validate the Portal site URLs
   d. If necessary, update the DNS mapping info.
      Once all of the APIs are verified, you may need to update the DNS mapping info, to redirect the calls to the new Gateways and Portals.

   The v10 migration is now complete.
4. After migration is finished, and verification of the target system is complete, you can uninstall API Connect on the source system.
   See Uninstalling API Connect.

Verifying the target system on VMware

Verify that the target system is in a healthy state and fully configured.

Before you begin

Complete Restoring the target system on VMware.

About this task

Check the health of the target system, complete any optional configuration, and validate data, APIs, and URLs.

Procedure

1. Login to your target management VM with **SSH**, and switch to the root user:

   ```
   ssh apicadm@<management hostname>
   ```

   ```
   sudo -i
   ```

   If you deployed a three replica management profile, then login to just one of the VMs.
2. Run the health check script:

   ```
   python3 health_check_post_migration.py -s <platform API endpoint> -u admin -p <password> -r admin/default-idp-1 -n default
   ```

   where
   - *<platform API endpoint>* is defined in the apiconnect-up-v10.yml file in your project directory.
   - *<password>* is the admin password that is used on your source system.

Migrating from OpenShift to Cloud Pak for Integration on the same cluster

Migrate your API Connect v10 deployment from OpenShift to Cloud Pak for Integration, while remaining on the same cluster.

Before you begin

Your API Connect instance must use the top-level CR. Direct migration of individual API Connect subsystem installations to Cloud Pak for Integration by the steps described here is not supported. Alternative methods of migration if you are not using the top-level CR are:

- If the same subsystem endpoints can be used, deploy a new top-level CR API Connect and restore disaster recovery backups from your existing installation, see Backing up, restoring, and disaster recovery. The new top-level CR deployment can then be migrated to Cloud Pak for Integration following the steps here.
- If different subsystem endpoints need to be used, migrate your API Connect deployment to a new top-level CR OpenShift cluster first. Follow the form factor migration steps Migrating from v10 to v10 on a different form factor, and then return here to complete the OpenShift top-level CR to Cloud Pak for Integration migration.

About this task

V10.0.6.0 Only: Migration from OpenShift to Cloud Pak for Integration on the same cluster results in changes to the Management subsystem API endpoint URIs, as shown in these examples.

- apiManager endpoint changes from

  `https://topcr-mgmt-api-manager-apic.example.com/manager`

  to

  `https://cpd-apic.example.com/integration/apis/apic/topcr/manager`

- cloudManager endpoint changes from

  `https://topcr-mgmt-admin-apic.example.com/admin`

  to

  `https://cpd-apic.example.com/integration/apis/apic/topcr/admin`

- platformAPI endpoint changes from

  `https://topcr-mgmt-platform-api-apic.example.com/`

  to

  `https://cpd-apic.example.com/integration/apis/apic/topcr/api`

- consumerAPI endpoint changes from

  `https://topcr-mgmt-consumer-api-apic.example.com/`

  to

  `https://cpd-apic.example.com/integration/apis/apic/topcr/consumer-api`

Procedure

1. Update your `<TOP_CR_NAME>-mgmt-admin-pass` secret with your current Cloud Manager admin password:
   a. Check whether the `<TOP_CR_NAME>-mgmt-admin-pass` password already matches your Cloud Manager admin password with:

   ```
   oc get secret -n <APIC-namespace> | grep mgmt-admin-pass

   oc get secret -n <APIC-namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -d &&
   echo ""
   ```

   If the passwords match then skip to 2.
   b. From where you run OpenShift CLI commands, use the base64 command to encode your Cloud Manager admin password.

   ```
   echo -n '<password>' | base64
   ```

   Note: If the `base64` command is not available, alternative commands or tools can be used.
   c. Edit the `<TOP_CR_NAME>-mgmt-admin-pass` secret:

   ```
   oc edit secret <TOP_CR_NAME>-mgmt-admin-pass -n <namespace>
   ```

   d. Replace the `data.password` value with your base64 encoded value from 1.b.

   ```
   ...
   data:
     email: ZXhhbXBsZUBleGFtcGxlLmNvbQ==
     password: <base64 encoded password>
   kind: Secret
   metadata:
   ...
   ```

   Note: The password in `<TOP_CR_NAME>-mgmt-admin-pass` is only used on API Connect installation, form factor migration, and disaster recovery. It does not need to be kept in sync with the Cloud Manager during normal API Connect usage.
2. Install IBM Cloud Pak Platform UI and create an instance of it.
3. Restart the `apiconnect` operator pod.
   The restarted operator pod detects Cloud Pak for Integration and reconciles the subsystems with it. Wait for the subsystems to be healthy.
4. Log in to Cloud Manager and check the services under the topology section.
   Known issue: If you see an additional Analytics service with a different name for the same Analytics endpoint, you must delete the newly created Analytics service.
5. Check that the configurator job is complete: :

```
oc get pods -n <namespace>
...
NAME                                                READY   STATUS      RESTARTS   AGE
<instance name>-configurator-<random string>        0/1     Completed   0          2h
...
```

Complete the following validation checks:

- Verify you are able to login to the Cloud Pak for Integration Platform UI, and can see your API Connect instance.
- Verify you are able to login to Cloud Manager UI, API Manager UI, and your portal sites.
- Test that your APIs are running, and analytics data is logged.
- Test lifecycle operations such as a product publish or delete, or an app creation.

v10 migration troubleshooting

Troubleshoot problems with v10 migration.

Topics:

- Script help
- Restarting migration
- Factory reset of portal subsystem
- Avoid error when a gateway is already registered to another API manager

Script help

```
python3 <SCRIPT_FILE_NAME> --help
```

Restarting migration

On the target system, after running **register_gateway_portals_in_target.py** or **update_to_new_portals.py** or **update_to_new_gateways.py** script:

- If you want to restore the Management database and try the steps again:
  - Run the restore Management db (Step 2 on Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration) with the **-reset_gateway_portal** flag as shown

    ```
    python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
    ```

    If running on a Kubernetes environment, supply the Cloud Manager admin password by using the **-p** flag. After this, continue with next steps of migration.
- If you deleted the data directory after registering Gateways or Portals, but want to use the same target API Connect system which you already installed, then:
  1. Complete Preparing the source system to create the data directory.
  2. Run the restore management db (Step 2 on Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration) with the **-reset_gateway_portal** flag as shown

     ```
     python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
     ```

     If running on a Kubernetes environment, supply the Cloud Manager admin password by using the **-p** flag. After this, continue with next steps of migration.

Factory reset of portal subsystem

You can do a factory reset of the Portal subsystem by either manually or by using a migration script.

- Using migration script
  The script will restore the Management database and will factory reset the Portal subsystem, and also delete Gateway pods. The gateway pods will get restarted.

  The script will factory reset the Portal subsystem and also delete Gateway pods. When Gateway pods are deleted, they are also restarted.

  ```
  python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
  ```

  If using a Kubernetes environment, you must also supply the Cloud Manager admin password by using the **-p** flag. After this command, you must run the remaining steps of migration, starting with Step 3 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.

  ```
  python3 register_gateway_portals_in_target.py -n NAMESPACE -u USERNAME -p PASSWORD -s HOSTNAME -r REALM_NAME -reset_gateway_portal
  ```

- **Manual steps**
  1. Log in to the Portal www pod:

     ```
     kubectl exec -it PORTAL_WWW_POD_NAME -c admin -n NAMESPACE -- bash
     ```

  2. Run the command :

     ```
     curl -X DELETE -k --cert /etc/nginx/ssl/portal-director-client-both.pem  https://localhost:3009/v2/portal/factory-reset
     ```

  3. If any sites are there, use the **list_sites** command to check if all sites are deleted.
  4. Check health of the Portal subsystem.

Avoid error when a gateway is already registered to another API manager

Error messages

```
Error: An error occurred communicating with the gateways subsystem at 'some_url' (status: 400, response: '"Unable to perform
initial registration for API Management.
Error: The API Connect Gateway Service is already registered to another API Manager instance"
```

You can fix the error condition manually, or use a script:

- Manual step, when Gateway is in a Kubernetes or OpenShift cluster:
  Restart all the gateway pods.
- Using the migration script when Gateway is in a Kubernetes or OpenShift cluster:
  The script will restore the Management database, factory reset the Portal subsystem, and delete the Gateway pods. Once deleted, Gateway pods get restarted.
  1. Run the following script:

     ```
     python3 restore_management_db.py -n namespace -s PLATFORM_API_HOSTNAME -reset_gateway_portal
     ```

     If using a Kubernetes environment, you must also supply the Cloud Manager admin password by using the **-p** flag.

  2. After this command, you must run the remaining steps of migration, starting with Step 3 in Restoring the target system on Kubernetes, OpenShift, and Cloud Pak for Integration.
- Gateway is on an Appliance
  Reboot all the Gateway appliances.

Shutting down the Postgres database

V10.0.6 only. Shutdown the Postgres database to ensure no changes are made to the database during migration.

Instructions:

- Shutting down Postgres database in Management subsystem on Kubernetes or VMware
- Shutting down Postgres database in Management subsystem on OpenShift

Shutting down Postgres database in Management subsystem on Kubernetes or VMware

When using s3 backup configuration for the Management subsystem, the following steps should be run on the source system, before the target API Connect system is installed.

1. If on a VMware appliance deployment, connect to the Management node and log in as root user (`sudo -i`), so that you can run `kubectl`.
2. Enable pgo client:

   ```
   kubectl set env deploy/ibm-apiconnect ENABLE_PGO_CLIENT="true" -n OPERATOR_NAMESPACE
   ```

3. Wait for `pgo-client` pod to come up.
4. Bring down the apic operator:

   ```
   kubectl scale deploy ibm-apiconnect --replicas=0 -n OPERATOR_NAMESPACE
   ```

5. Scale down deployments.
   Before scaling down the deployments, make a note of the current replicas:

   ```
   kubectl get deployment -n namespace
   ```

   - Standard deployments :
     - juhu
     - portal-proxy
     - apim
     - lur
     - taskmanager
     - hub (when Test and Monitor is enabled)

   a. Scale down:

   ```
   kubectl scale deploy <deployment_name> --replicas=0 -n NAMESPACE
   ```

   b. To get the name of the Postgres cluster:

   ```
   kubectl get pgcluster -n NAMESPACE
   ```

   c. Connect to pgo client pod and run:

   ```
   pgo update cluster PG_CLUSTER_NAME --shutdown -n NAMEPSACE
   ```

Shutting down Postgres database in Management subsystem on OpenShift

When using s3 configuration, the following steps should be run on the source system, before the target API Connect system is installed.

1. Change to operator namespace:

   ```
   oc project "{OPERATOR NS}"
   ```

2. `oc edit csv ibm-apiconnect.v2.4.0`
   a. Search for `APPLIANCE` and add these two lines above it:

   ```
   - name: ENABLE_PGO_CLIENT
     value: "true"
   ```

b. Save and wait for the pgo-client pod to come up.

3. `oc scale deploy ibm-apiconnect --replicas=0`
4. Bring down apiconnect operator pod:

   ```
   oc scale deploy ibm-apiconnect --replicas=0
   ```

5. Change to the namespace where API Connect is installed:

   ```
   oc project "{APIC NS}"
   ```

6. Scale down the following deployments: juhu, portal-proxy, apim, lur, taskmanager

   ```
   oc scale deploy JUHU_DEPLOYMENT_NAME --replicas=0
   oc scale deploy PORTAL_PROXY_DEPLOYMENT_NAME --replicas=0
   oc scale deploy APIM_DEPLOYMENT_NAME --replicas=0
   oc scale deploy LUR_DEPLOYMENT_NAME --replicas=0
   oc scale deploy TASKMANAGER_DEPLOYMENT_NAME --replicas=0
   ```

7. To get the name of the pg cluster:

   ```
   oc get pgcluster
   ```

8. Log in to pgo client pod:

   ```
   oc exec -it $(oc get po|grep pgo|awk '{print $1}') -- bash
   ```

9. Shutdown Postgres database:

   ```
   pgo update cluster PG_CLUSTER_NAME --shutdown -n NAMESPACE
   ```

Starting the Postgres database in the Management subsystem on Kubernetes and OVA

Use these steps when you are moving back to the source system because the migration failed, and you want to bring up the Postgres database, if it was shutdown.

1. If on an appliance (VMware), connect to the Management node and log in as root user (`sudo -i`), so that you can run the `kubectl` commands.
2. Get the name of the Postgres cluster:

   ```
   kubectl get pgcluster -n NAMESPACE
   ```

3. Connect to the pgo client pod and run the command:

   ```
   pgo update cluster PG_CLUSTER_NAME --startup -n NAMESPACE
   ```

4. Scale up the deployments that use Postgres.
   Standard deployments : juhu, portal-proxy, apim, lur, taskmanager, hub (when test and monitor is enabled)

   ```
   kubectl scale deploy DEPLOYMENT_NAME --replicas=N -n NAMEPSACE
   ```

   The value of `N` is from the `kubectl get deployment -n namespace` output that you got before shutting down the Postgres database. It is 1 if using n1 profile, and 3 when using n3 profile.

5. Bring up the apic operator:

   ```
   kubectl scale deploy ibm-apiconnect --replicas=1 -n OPERATOR_NAMESPACE
   ```

6. If s3 backups are used, perform a Management restore with the backup that was taken as part of Preparing the source system.
   Note: Any backups that are taken on the target system are not valid for the source system. The backups must not be used during restoration.
7. Ensure that all subsystems are healthy.

Starting postgres database on the Management subsystem on OpenShift

Use these steps when you are moving back to the source system because the migration failed, and you want to bring up the Postgres database, if it was shutdown.

1. Change to the project where the top-level CR (apic) is installed:

   ```
   oc project APIC_NAMESPACE
   ```

2. Get the name of the Postgres cluster:

   ```
   oc get pgcluster
   ```

3. Log in to pgo client pod:

   ```
   oc exec -it $(oc get po|grep pgo|awk '{print $1}') -- bash
   ```

4. Start the postgres database:

   ```
   pgo update cluster PG_CLUSTER_NAME --startup -n NAMESPACE
   ```

5. Scale up the deployments: juhu, portal-proxy, apim, lur, and taskmanager. Use the following commands, where `N` is the value of replicas. If an n1 profile being used the value is 1, and if an n3 profile is being used the value is 3.
   - `oc scale deploy JUHU_DEPLOYMENT_NAME --replicas=N`
   - `oc scale deploy PORTAL_PROXY_DEPLOYMENT_NAME --replicas=N`
   - `oc scale deploy APIM_DEPLOYMENT_NAME --replicas=N`
   - `oc scale deploy LUR_DEPLOYMENT_NAME --replicas=N`
   - `oc scale deploy TASKMANAGER_DEPLOYMENT_NAME --replicas=N`
6. Bring down the apiconnect operator pod:

   ```
   oc scale deploy ibm-apiconnect --replicas=1 -n OPERATOR_NAMESPACE
   ```

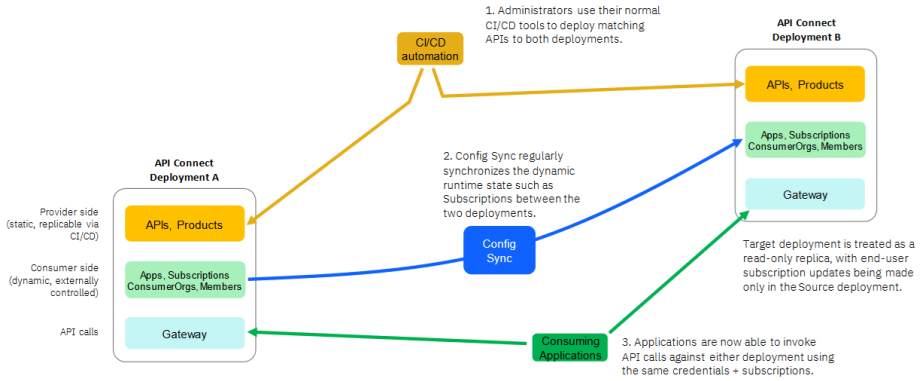
7. If s3 backups are used, perform a Management restore with the backup that was taken as part of Preparing the source system.
   Note: Any backups that are taken on the target system are not valid for the source system. The backups must not be used during restoration.
8. Ensure that all subsystems are healthy.

[Technical Preview] Using Config Sync to replicate consumer-side catalog data

Use the API Connect Config Sync utility to replicate consumer-side data from a source catalog in an API Connect cluster to a target catalog in a different API Connect cluster, so that API calls can be served by either API Connect cluster interchangeably to enable hybrid deployments and high-availability style scenarios.

API Connect Config Sync (apic-configsync) can be run either as a standalone binary, or a schedule with a Kubernetes cronjob, to facilitate the unidirectional replication of consumer-side data (consumer organizations, members, apps, subscriptions, credentials) from a catalog in a source API Connect cluster to a corresponding catalog in a separate, target API Connect cluster.

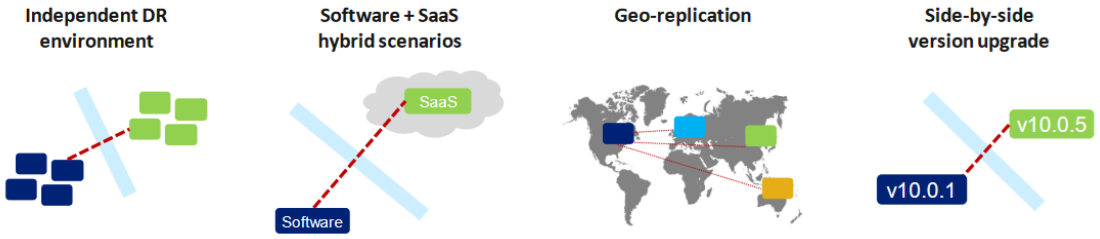
The following image shows how you can use Config Sync to replicate consumer-side data from a source deployment to a target deployment:



Config Sync can be run as frequently as required, and detects the differences between the replicated resource types in the source and target catalogs so that it efficiently replicates only the new changes that have been made in the source instance.

As shown in the following image, Config Sync helps to enable a range of hybrid cloud API topologies such as:

- Independent disaster recovery instances
- Software and SaaS hybrid scenarios
- Geo-replication
- Side-by-side version upgrade



This feature is provided as a Technical Preview

As such, the feature might not be fully functional in your environment and support is limited. In addition, the feature's appearance and functionality are subject to change both in this release and in later releases of API Connect. For more information, see https://ibm.biz/api-connect-config-sync.

Requirements

For API Connect Config Sync to run successfully, the following requirements that must be satisfied:

- You are responsible for configuring the API Connect the target deployment to appropriately match the source deployment, so that replication of consumer-side data can take place:
  - The target API Connect deployment must be configured correctly including topology setup, provider organization creation, and catalog creation.
  - The APIs and Products in the source catalog must be replicated in the target catalog using either CI/CD or manual procedures.
  - The target catalog settings (including any spaces if they are enabled in the source catalog) must be configured to match the configuration of the source catalog. This includes compatible gateways and any extensions, user registries, and any other relevant catalog settings (such as lifecycle approvals, application lifecycle, and so on).
  - Compatible onboarding User Registries must be configured in the target catalog (the user registry name and type must match that of the corresponding source user registry, that is, the source and target catalog names must match if you are using the default user registry).
  - All OAuth providers must be configured on the target catalog and spaces.
  - All API User Registries must be configured on the target catalog and spaces.
  - All TLS client profiles must be configured on the target catalog and spaces.
- The Config Sync utility makes use of the API Connect Provider REST API, so you must have connectivity (into both the source and target clusters) with sufficient provider organization user permissions.
- The `metadata.source_id` key is reserved for use by Config Sync.

    - Because the underlying creation of resources in the target catalog uses the API Connect Provider REST API, the IDs and URLs associated with resources in the target catalog will be different from their corresponding resources in the source catalog (only the cluster-agnostic data of a resource is synchronized).
    - To track which resources in the source catalog map to which resources in the target catalog, Config Sync uses the `metadata` field of a particular resource. When a new resource is created in the target catalog, the `metadata` field of that resource is set to contain a key named `source_id` with a value equal to the ID of the corresponding resource in the source catalog that is being synchronized.
- API Connect Config Sync also uses the `name` field of some resources to determine a match. This is typically done for resources that are not synchronized by the utility but are needed in the target catalog for the synchronization to be successful; for example, User Registries, Consumer Org Groups, Roles, and published Products (the `name` and `version` fields are used for product matching).

Note: The first time API Connect Config Sync runs, it might take longer to complete than subsequent runs require (depending on the number of resources to synchronize). The first run of the utility creates all new resources in the target cluster; subsequent runs only perform synchronization of any data that was added, updated, or deleted since the last run of the utility.

Limitations

API Connect Config Sync has the following limitations:

- Config Sync does not back up data; it merely replicates the contents of a source catalog to a target catalog. You must still perform periodic backups to ensure the safety of your data.
- The target catalog is a read-only replica: only unidirectional synchronization of data from the source catalog to the target catalog is supported. Subsequent changes made in the target catalog cannot be synchronized back to the source.
- Source and target deployments operate independently of each other:
    - API rate limiting on Plans is counted independently in the source catalog and the target catalog.
    - OIDC tokens are not replicated. OIDC tokens generated in the source cluster are not valid within the target cluster.
    - Local User Registry (LUR) passwords are not replicated. Consumer organization members belonging to a Local User Registry (LUR) must reset their password in the target cluster.
    - Analytics data is tracked independently in the source cluster and the target cluster. Analytics data is not replicated or aggregated across clusters.
    - Developer portal configuration is not replicated. If you want to enable consumer organization access in the target catalog, you must replicate your Developer Portal site for consistency (restore the site with UUID/URL update where needed).
- Config Sync operates on a single source-target catalog pair at a time.
  If you want to sync multiple different source-target catalog pairs, create a separate cronjob for each pair.

Supported versions of API Connect

Config Sync can synchronize consumer-side resources across different versions of API Connect clusters. The target API Connect cluster version must always be greater or equal to the source API Connect cluster version. Table 1 shows the supported version combinations:

Table 1. Supported source and target version combinations of API Connect

Source version	Target Version	Release Stream
10.0.1.15	10.0.5+	Earlier LTS -> Current LTS
10.0.5+	10.0.5+	Current LTS -> Current LTS
10.0.5+	10.0.7	Current LTS -> Current CD
10.0.7	10.0.7	Current CD -> Current CD

- **Downloading the API Connect Config Sync utility**
  Download and extract the API Connect Config Sync utility from the Early Access Program site.
- **Setting environment variables for Config Sync**
  Configure environment variables that can be used by the API Connect Config Sync utility.
- **Running Config Sync as a standalone binary**
  The pre-built binaries allow you to run the API Connect Config Sync utility from a Mac, Windows, or Linux x86 platform.
- **Running Config Sync as a cronjob on OpenShift**
  Running the apic-configsync utility as a cronjob on Red Hat OpenShift or IBM Cloud Pak for Integration enables the utility to be run on a timed basis, such as every 10 minutes.
- **Running Config Sync as a cronjob on Kubernetes**
  Running the apic-configsync utility as a cronjob on Kubernetes enables the utility to be run on a timed basis, such as every 10 minutes.
- **Running Config Sync as a cronjob on VMware**
  Running the apic-configsync utility as a cronjob on VMware enables the utility to be run on a timed basis, such as every 10 minutes.

Downloading the API Connect Config Sync utility

Download and extract the API Connect Config Sync utility from the Early Access Program site.

About this task

The apic-configsync.tar.gz file contains the following artifacts:

- A binaries directory with sub-directories containing the apic-configsync utility for each of the supported client platforms (Mac, Windows, and Linux x86).
- A helper_files directory containing YAML files for configuring cronjobs if you want to run the utility on a schedule.
- An ibm-apic-configsync-image.tar.gz file containing the Docker image for running the utility on a schedule in a Kubernetes cluster with a cronjob.

Procedure

1. Sign up to the API Connect Config Sync Early Access Program at the following location: https://ibm.biz/api-connect-config-sync.

2. Agree to the terms.
3. Download the apic-configsync.tar.gz file.
4. Extract the apic-configsync.tar.gz package by running the following command:

```
tar xf apic-configsync.tar.gz
```

Setting environment variables for Config Sync

Configure environment variables that can be used by the API Connect Config Sync utility.

API Connect Config Sync consumes a set of configuration properties required to connect to and perform REST calls against the source and target API Connect clusters. The configuration properties are passed to the utility with environment variables.

The following environment variables must be set to the appropriate values for the source and target API Connect cluster:

```
# Properties for source catalog
export SOURCE_MGMT_SERVER=https://yoursourcehostname.com/path
export SOURCE_ORG=sourceporg
export SOURCE_REALM=provider/default-idp-2
export SOURCE_ADMIN_USER=sourceadmin
export SOURCE_ADMIN_PASSWORD=sourceadminpassword
#export SOURCE_ADMIN_APIKEY=1010101010101010
export SOURCE_CATALOG=sourcecatalog
export SOURCE_TOOLKIT_CREDENTIALS_CLIENTID=00000000-0000-0000-0000-000000000000
export SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET=11111111-1111-1111-1111-111111111111

# Properties for target catalog
export TARGET_MGMT_SERVER=https://yourtargethostname.com/path
export TARGET_ORG=targetporg
export TARGET_REALM=provider/default-idp-2
export TARGET_ADMIN_USER=targetadmin
export TARGET_ADMIN_PASSWORD=targetadminpassword
#export TARGET_ADMIN_APIKEY=2020202020202020
export TARGET_CATALOG=targetcatalog
export TARGET_TOOLKIT_CREDENTIALS_CLIENTID=22222222-2222-2222-2222-222222222222
export TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET=33333333-3333-3333-3333-333333333333
```

Download the Toolkit credentials file

A number of the required configuration properties can be obtained from the Toolkit credentials file that can be downloaded from the API manager UI for the API Connect cluster.

1. Log in to the API Manager UI.
2. From the Home page, select the Download Toolkit tile.
3. In "Step 2" of the Download dialog box, select Toolkit credentials - Download.
4. Open the resulting credentials.json file in a text editor so that you can locate and copy settings as explained in the next section.

Locating information for the environment variables

The following list explains how to obtain the setting for each of the environment variables.

`*_MGMT_SERVER`
: This property specifies the hostname and path that provides access to the REST API endpoint for your catalog. Locate **`toolkit.endpoint`** property in the toolkit **`credentials.json`** file, and set the **`*_MGMT_SERVER`** variable to the value of the **`toolkit.endpoint`** property.
The value might look like the following example:

  ```
  https://platform-api.hostname.com/api
  ```

`*_TOOLKIT_CREDENTIALS_CLIENTID, *_TOOLKIT_CREDENTIALS_CLIENTSECRET`
: The toolkit credentials **`clientID`** and **`clientSecret`** are used to authenticate the utility to be able to make REST API calls against the API Connect cluster. Locate following properties in the toolkit **`credentials.json`** file, and set the environment variables as shown:

  - Set the **`*_TOOLKIT_CREDENTIALS_CLIENTID`** variable to the value of the **`toolkit.client_id`** property
  - Set the **`*_TOOLKIT_CREDENTIALS_CLIENTSECRET`** variable to the value of the **`toolkit.client_secret`** property

`*_ORG`
: This is the short name of the provider organization in which your catalog is hosted. Set the **`*_ORG`** variable to the value of the provider organization's **`Name`** property. To locate the provider organization name, complete the following steps:

  1. Log in to the API Manager.
  2. From the Home tab, select Manage settings.
  3. On the Settings page, click the Overview tab and locate the value of the **`Name`** property (not the **`Title`**).

`*_REALM`
: This property specifies the identity provider realm of the API Manager registry for your instance of API Connect. Set the **`*_REALM`** variable to the value of the realm name. This parameter is not required in API Connect instances with single-sign-on (SSO) enabled on the API Manager user registry (for example, API Connect Enterprise as a Service on AWS). Instead, you authenticate using an API key (see **`*_API_KEY`** for details).
To obtain the realm name, run the following toolkit CLI command:

  ```
  apic identity-providers:list --scope provider --server <mgmt_endpoint_url> --fields title,realm
  ```

  where ***`<mgmt_endpoint_url>`*** is the API Manager URL with any segments (for example, **`/manage`**) removed from the end of the URL. If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in Viewing platform and UI endpoints.

The following example shows a response listing the available realms for a deployment:

```
apic identity-providers:list --scope provider --server <mgmt_endpoint_url> --fields title,realm

total_results: 2
results:
  - title: 'API Manager User Registry'
    realm: 'provider/default-idp-2'
  - title: 'Common Services User Registry'
    realm: 'provider/common-services'
```

In the example, the realm for the API Manager registry is **provider/default-idp-2**.

For more information on obtaining the realm name, see How to determine the identity provider.

***_CATALOG**
This property indicates the name of the Catalog that you want to synchronize. Set the ***_CATALOG** variable to the value of the catalog's **Name** property.
To locate the catalog name, complete the following steps:

1. Log in to the API Manager UI.
2. From the Home tab, select Manage catalogs.
3. Select the tile for the catalog you want to synchronize.
4. Click the Catalog settings tab.
5. On the Settings page, locate the value of the catalog's **Name** property (not the **Title**).

***_ADMIN_USER**
This property is the name of the user account under which you will authenticate to the catalog. Set the ***_ADMIN_USER** variable to the value of the username that the utility will log in with to perform the synchronization task. This parameter is not required in API Connect instances with single-sign-on (SSO) enabled on the API Manager user registry (for example, API Connect Enterprise as a Service on AWS). Instead, you authenticate using an API key (see ***_API_KEY** for details).
In the source catalog, the user must be a Catalog Administrator or Organization Administrator with the ability to create consumer objects on behalf of the end users.
In the target catalog, the user must have authorization to Update the catalog settings and enable or disable the **hash_client_secret** property.

To confirm the username, complete the following steps:

1. Log in to the API Manager UI with the user account you will use to synchronize the catalog.
2. Click the username in the page header.
3. Select My account.
4. Locate the value of the Username property.

***_ADMIN_PASSWORD**
In most cases you will authenticate to the API Manager with the password associated to your username. Set the ***_ADMIN_PASSWORD** property to your password, being careful with any non-alphanumeric characters that may be in your password. This parameter is not required in API Connect instances with single-sign-on (SSO) enabled on the API Manager user registry (for example, API Connect Enterprise as a Service on AWS). Instead, you authenticate using an API key (see ***_API_KEY** for details).
Attention: If you set the ***_ADMIN_PASSWORD** variable, then skip the ***_ADMIN_APIKEY** variable. Do not set both variables when synchronizing a particular catalog.

***_ADMIN_APIKEY**
In API Connect instances with single-sign-on (SSO) enabled on the API Manager user registry (for example API Connect Enterprise as a Service), you must authenticate using an API key instead of your password.
To locate your API key, complete the following steps:

1. Log in to the API Manager UI using your IBM ID.
2. In the page header, click and select My API Keys.
3. On the My API Keys page, click Add.
4. Set a title and description for your new API key and click Create.
5. Copy the new API key from the dialog box.

Attention: If you set the ***_ADMIN_APIKEY** variable, then skip the ***_ADMIN_PASSWORD** variable. Do not set both variables when synchronizing a particular catalog.

Setting the environment variables

- Mac and Linux x86:
  On Mac or Linux x86, you can either export the environment variables individually on the command line directly or create a script to export the environment variables and source that file from the command line, as in the following example:

```
# Properties for source catalog
export SOURCE_MGMT_SERVER=https://yoursourcehostname.com/path
export SOURCE_ORG=sourceporg
export SOURCE_REALM=provider/default-idp-2
export SOURCE_ADMIN_USER=sourceadmin
export SOURCE_ADMIN_PASSWORD=sourceadminpassword
#export SOURCE_ADMIN_APIKEY=1010101010101010
export SOURCE_CATALOG=sourcecatalog
export SOURCE_TOOLKIT_CREDENTIALS_CLIENTID=00000000-0000-0000-0000-000000000000
export SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET=11111111-1111-1111-1111-111111111111

# Properties for target catalog
export TARGET_MGMT_SERVER=https://yourtargethostname.com/path
export TARGET_ORG=targetporg
export TARGET_REALM=provider/default-idp-2
export TARGET_ADMIN_USER=targetadmin
export TARGET_ADMIN_PASSWORD=targetadminpassword
#export TARGET_ADMIN_APIKEY=2020202020202020
export TARGET_CATALOG=targetcatalog
export TARGET_TOOLKIT_CREDENTIALS_CLIENTID=22222222-2222-2222-2222-222222222222
export TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET=33333333-3333-3333-3333-333333333333
```

In the example, notice that the command to set the ***_ADMIN_APIKEY** variable is commented out because the ***_ADMIN_PASSWORD** is used instead.

- Windows:
  On Windows, you can set each environment variable individually by opening a command window and running the following command for each variable:

  ```
  setx <variable_name> "<variable_value>"
  ```

  where:
  - ***<variable_name>*** is the name of the environment variable to set
  - ***<variable_value>*** is the value to set for the specified variable name

  Alternatively, you can create a batch file that uses the **setx** command to set the environment variables, as in the following example:

  ```
  REM Properties for source catalog
  setx SOURCE_MGMT_SERVER "https://yoursourcehostname.com/path"
  setx SOURCE_ORG "sourceporg"
  setx SOURCE_REALM "provider/default-idp-2"
  setx SOURCE_ADMIN_USER "sourceadmin"
  setx SOURCE_ADMIN_PASSWORD "sourceadminpassword"
  REM setx SOURCE_ADMIN_APIKEY "1010101010101010"
  setx SOURCE_CATALOG "sourcecatalog"
  setx SOURCE_TOOLKIT_CREDENTIALS_CLIENTID "00000000-0000-0000-0000-000000000000"
  setx SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET "11111111-1111-1111-1111-111111111111"

  REM Properties for target catalog
  setx TARGET_MGMT_SERVER "https://yourtargethostname.com/path"
  setx TARGET_ORG "targetporg"
  setx TARGET_REALM "provider/default-idp-2"
  setx TARGET_ADMIN_USER "targetadmin"
  setx TARGET_ADMIN_PASSWORD "targetadminpassword"
  REM setx TARGET_ADMIN_APIKEY "2020202020202020"
  setx TARGET_CATALOG "targetcatalog"
  setx TARGET_TOOLKIT_CREDENTIALS_CLIENTID "22222222-2222-2222-2222-222222222222"
  setx TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET "33333333-3333-3333-3333-333333333333"
  ```

  In the example batch file, notice that the command to set the ***_ADMIN_APIKEY** variable is commented out because the ***_ADMIN_PASSWORD** is used instead.

Running Config Sync as a standalone binary

The pre-built binaries allow you to run the API Connect Config Sync utility from a Mac, Windows, or Linux x86 platform.

About this task

Running a standalone invocation of apic-configsync using one of the pre-built binaries is the most flexible option. For optimal performance, apic-configsync should run as close to the target cluster as possible.

Complete the following steps to configure environment variables and run the standalone binary.

Procedure

1. Download and extract the API Connect Config Sync package.
2. In the extracted package, navigate to the /binaries/*<platform>* directory that contains the appropriate version of the apic-configsync utility for your platform.
3. Define the environment variables for the source and target catalogs.
4. Run the utility with the following command:
   - Mac and Linux x86: `./apic-configsync`
   - Windows: `apic-configsync.exe`

Running Config Sync as a cronjob on OpenShift

Running the apic-configsync utility as a cronjob on Red Hat OpenShift or IBM Cloud Pak for Integration enables the utility to be run on a timed basis, such as every 10 minutes.

Before you begin

Ensure that the following prerequisites are satisfied before attempting to configure the cronjob:

- Ensure that you have a tool available for copying the image into your image registry.
  The examples in this topic use **skopeo** to copy the image to the target registry, but you could use equivalent **docker** or **podman** commands instead.
- Expose the OpenShift registry with **defaultRoute** by running the following command:

  ```
  oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":{"defaultRoute":true}}' --type=merge
  ```

  For more information, see Exposing the registry in the OpenShift documentation. It is not necessary to perform step 5 (the "podman login" step) in the instructions.
- Set the **REGISTRY_HOST** environment variable to the OpenShift registry by running the following command:

  ```
  REGISTRY_HOST=$(oc get route default-route -n openshift-image-registry --template='{{`{{ .spec.host }}`}}')
  ```

- Set the **APIC_PROJECT** environment variable to the namespace (project) where you intend to run the cronjob. For optimal performance, the cronjob should run within the same OpenShift environment as the target API Connect cluster.

About this task

You can configure the apic-configsync utility to run as a cronjob in an API Connect deployment running on OpenShift or Cloud Pak for Integration. The apic-configsync utility should run as close to the target cluster as possible, so the cronjob should be configured to run within the same OpenShift environment as the target API Connect cluster.

Procedure

1. Download and extract the API Connect Config Sync package.
2. Extract the Config Sync image from the extracted API Connect Config Sync package by running the following command:

   ```
   tar xf ibm-apic-configsync-image.tar.gz
   ```

3. Copy the extracted Config Sync image to the target registry.
   For example, if you use skopeo, run the following command:

   ```
   skopeo copy --dest-tls-verify=false --dest-registry-token=$(oc whoami -t) dir:./ibm-apic-configsync/ibm-apic-configsync-<version>/<tag> docker://$REGISTRY_HOST/$APIC_PROJECT/ibm-apic-configsync-<version>:<tag>
   ```

   Replace ***<version>*** and ***<tag>*** with their actual values. To ensure that you use the most recent values, copy them from the README file located in the root of the extracted API Connect Config Sync package.

4. Define the environment variables for the source and target catalogs.
5. Create secrets to host configuration details for the source cluster and the target cluster.
   a. Create a secret to host the configuration details for the source cluster in the namespace where you intend to run the cronjob:

   ```
   oc create secret --namespace $APIC_PROJECT generic source-catalog-details \
       --from-literal=SOURCE_MGMT_SERVER=$SOURCE_MGMT_SERVER \
       --from-literal=SOURCE_ORG=$SOURCE_ORG \
       --from-literal=SOURCE_REALM=$SOURCE_REALM \
       --from-literal=SOURCE_ADMIN_USER=$SOURCE_ADMIN_USER \
       --from-literal=SOURCE_ADMIN_PASSWORD=$SOURCE_ADMIN_PASSWORD \
       --from-literal=SOURCE_CATALOG=$SOURCE_CATALOG \
       --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTID=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTID \
       --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET
   ```

   b. Create a secret to host the configuration details for the target cluster in the namespace where you intend to run the cronjob:

   ```
   oc create secret --namespace $APIC_PROJECT generic target-catalog-details \
       --from-literal=TARGET_MGMT_SERVER=$TARGET_MGMT_SERVER \
       --from-literal=TARGET_ORG=$TARGET_ORG \
       --from-literal=TARGET_REALM=$TARGET_REALM \
       --from-literal=TARGET_ADMIN_USER=$TARGET_ADMIN_USER \
       --from-literal=TARGET_ADMIN_PASSWORD=$TARGET_ADMIN_PASSWORD \
       --from-literal=TARGET_CATALOG=$TARGET_CATALOG \
       --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTID=$TARGET_TOOLKIT_CREDENTIALS_CLIENTID \
       --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET=$TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET
   ```

6. Create the cronjob:
   a. Open the `helper_files/configsync-cronjob-ocp.yaml` file located in the extracted API Connect Config Sync package.
   b. Modify the `schedule` property to specify how frequently the job will run.
   The minimum recommended schedule period is 1 minute; however, a deployment with a large number of objects might require more time for the job to complete. Because concurrent jobs are not allowed, the total time between the start of two consecutive jobs might be longer.

   c. Set the `concurrencyPolicy` to `Forbid`.
   Config Sync operates on a single source-target catalog pair at a time. If you want to sync multiple different source-target catalog pairs, create a separate cronjob for each pair.

   d. In the `image` property, replace the ***<APIC_PROJECT>*** variable with the namespace (project) where you will run the cronjob.
   e. Create the cronjob resource in the cluster by running the following command:

   ```
   oc --namespace $APIC_PROJECT apply -f helper_files/configsync-cronjob-ocp.yaml
   ```

Running Config Sync as a cronjob on Kubernetes

Running the apic-configsync utility as a cronjob on Kubernetes enables the utility to be run on a timed basis, such as every 10 minutes.

Before you begin

Ensure that the following prerequisites are satisfied before attempting to configure the cronjob:

- Ensure that you have a tool available for copying the image into your image registry.
  The examples in this topic use `skopeo` to copy the image to the target registry, but you could use equivalent `docker` or `podman` commands instead.

About this task

You can configure the apic-configsync utility to run as a cronjob in an API Connect deployment running on Kubernetes. The apic-configsync utility should run as close to the target cluster as possible, so the cronjob should be configured to run within the same Kubernetes environment as the target API Connect cluster.

Procedure

1. Download and extract the API Connect Config Sync package.
2. Extract the Config Sync image from the extracted API Connect Config Sync package by running the following command:

   ```
   tar xf ibm-apic-configsync-image.tar.gz
   ```

3. Copy the extracted Config Sync image to the target registry.
   For example, if you use skopeo, run the following command:

   ```
   skopeo copy dir:./ibm-apic-configsync/ibm-apic-configsync-<version>/<tag> docker://<registry>/ibm-apic-configsync/ibm-apic-configsync-<version>:<tag>
   ```

   Replace ***<version>*** and ***<tag>*** with their actual values. To ensure that you use the most recent values, copy them from the README file located in the root of the extracted API Connect Config Sync package.

   Replace ***<registry>*** with the actual registry value for your image registry. Refer to the skopeo documentation on configuring credentials for your destination registry.

4. Define the environment variables for the source and target catalogs.
5. Create secrets to host configuration details for the source cluster and the target cluster.
   a. Create a secret to host the configuration details for the source cluster in the namespace where you intend to run the cronjob:

   ```
   kubectl create secret --namespace <namespace> generic source-catalog-details \
       --from-literal=SOURCE_MGMT_SERVER=$SOURCE_MGMT_SERVER \
       --from-literal=SOURCE_ORG=$SOURCE_ORG \
       --from-literal=SOURCE_REALM=$SOURCE_REALM \
       --from-literal=SOURCE_ADMIN_USER=$SOURCE_ADMIN_USER \
       --from-literal=SOURCE_ADMIN_PASSWORD=$SOURCE_ADMIN_PASSWORD \
       --from-literal=SOURCE_CATALOG=$SOURCE_CATALOG \
       --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTID=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTID \
       --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET
   ```

   b. Create a secret to host the configuration details for the target cluster in the namespace where you intend to run the cronjob:

   ```
   kubectl create secret --namespace <namespace> generic target-catalog-details \
       --from-literal=TARGET_MGMT_SERVER=$TARGET_MGMT_SERVER \
       --from-literal=TARGET_ORG=$TARGET_ORG \
       --from-literal=TARGET_REALM=$TARGET_REALM \
       --from-literal=TARGET_ADMIN_USER=$TARGET_ADMIN_USER \
       --from-literal=TARGET_ADMIN_PASSWORD=$TARGET_ADMIN_PASSWORD \
       --from-literal=TARGET_CATALOG=$TARGET_CATALOG \
       --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTID=$TARGET_TOOLKIT_CREDENTIALS_CLIENTID \
       --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET=$TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET
   ```

6. Create the cronjob:
   a. Open the `helper_files/configsync-cronjob-k8s.yaml` file located in the extracted API Connect Config Sync package.
   b. Modify the `schedule` property to specify how frequently the job will run.
   The minimum recommended schedule period is 1 minute; however, a deployment with a large number of objects might require more time for the job to complete. Because concurrent jobs are not allowed, the total time between the start of two consecutive jobs might be longer.

   c. Set the `concurrencyPolicy` to `Forbid`.
   Config Sync operates on a single source-target catalog pair at a time. If you want to sync multiple different source-target catalog pairs, create a separate cronjob for each pair.

   d. In the `image` property, replace the ***<registry>*** variable with the actual registry value for your image registry.
   e. Create the cronjob resource in the cluster by running the following command:

   ```
   kubectl --namespace <namespace> apply -f helper_files/configsync-cronjob-k8s.yaml
   ```

Running Config Sync as a cronjob on VMware

Running the apic-configsync utility as a cronjob on VMware enables the utility to be run on a timed basis, such as every 10 minutes.

About this task

You can configure the apic-configsync utility to run as a cronjob in an API Connect deployment running on VMware. The apic-configsync utility should run as close to the target cluster as possible, so the cronjob should be configured to run from the management subsystem virtual machines in the target API Connect cluster.

Ensure that your virtual machines have enough resources available (CPU/Memory) for the cronjob to run the apic-configsync utility.

Procedure

1. Download and extract the API Connect Config Sync package.
2. Define the environment variables for the source and target catalogs.
3. For each node in the management subsystem:
   a. In the extracted apic-configsync.tar.gz package, locate the `ibm-apic-configsync-image.tar.gz` file.
   b. Upload the `ibm-apic-configsync-image.tar.gz` file using the following command:

   ```
   scp ibm-apic-configsync-image.tar.gz apicadm@<hostname>:~
   ```

   where ***<hostname>*** is the hostname or IP address of the node.

   c. SSH into the node and become root with `sudo -i`.
   d. Switch into the registry directory with the following command:

```
cd /var/lib/apiconnect-subsystem/registry/
```

e. Extract the image from the tar.gz file with the following command:

```
tar xf /home/apicadm/ibm-apic-configsync-image.tar.gz
```

Extracting the file creates the following directory:

```
/var/lib/apiconnect-subsystem/registry/ibm-apic-configsync/ibm-apic-configsync-<version>/<tag>
```

where ***<version>*** is the current version of the apic-configsync utility and ***<tag>*** is the tag of the built image.

f. Restart the appliance-manager by running the following command:

```
systemctl restart appliance-manager
```

4. Create secrets to host configuration details for the source cluster and the target cluster.
   a. SSH into one of the nodes in the management subsystem and become root with `sudo -i`.
   b. Create a secret to host the configuration details for the source cluster in the namespace where you intend to run the cronjob:

```
kubectl create secret generic source-catalog-details \
    --from-literal=SOURCE_MGMT_SERVER=$SOURCE_MGMT_SERVER \
    --from-literal=SOURCE_ORG=$SOURCE_ORG \
    --from-literal=SOURCE_REALM=$SOURCE_REALM \
    --from-literal=SOURCE_ADMIN_USER=$SOURCE_ADMIN_USER \
    --from-literal=SOURCE_ADMIN_PASSWORD=$SOURCE_ADMIN_PASSWORD \
    --from-literal=SOURCE_CATALOG=$SOURCE_CATALOG \
    --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTID=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTID \
    --from-literal=SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET=$SOURCE_TOOLKIT_CREDENTIALS_CLIENTSECRET
```

   c. Create a secret to host the configuration details for the target cluster in the namespace where you intend to run the cronjob:

```
kubectl create secret generic target-catalog-details \
    --from-literal=TARGET_MGMT_SERVER=$TARGET_MGMT_SERVER \
    --from-literal=TARGET_ORG=$TARGET_ORG \
    --from-literal=TARGET_REALM=$TARGET_REALM \
    --from-literal=TARGET_ADMIN_USER=$TARGET_ADMIN_USER \
    --from-literal=TARGET_ADMIN_PASSWORD=$TARGET_ADMIN_PASSWORD \
    --from-literal=TARGET_CATALOG=$TARGET_CATALOG \
    --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTID=$TARGET_TOOLKIT_CREDENTIALS_CLIENTID \
    --from-literal=TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET=$TARGET_TOOLKIT_CREDENTIALS_CLIENTSECRET
```

5. Create the cronjob:
   a. Open the `helper_files/configsync-cronjob-ova.yaml` file located in the extracted API Connect Config Sync package.
   b. Modify the `schedule` property to specify how frequently the job will run.
   The minimum recommended schedule period is 1 minute; however, a deployment with a large number of objects might require more time for the job to complete. Because concurrent jobs are not allowed, the total time between the start of two consecutive jobs might be longer.

   c. Set the `concurrencyPolicy` to `Forbid`.
   Config Sync operates on a single source-target catalog pair at a time. If you want to sync multiple different source-target catalog pairs, create a separate cronjob for each pair.

   d. Upload the `helper_files/configsync-cronjob-ova.yaml` file to one of the management nodes.
   To upload the file, run the following command:

```
scp helper_files/configsync-cronjob-ova.yaml apicadm@<hostname>:~
```

   Replace ***<hostname>*** with the host name or IP of one of the nodes in the management subsystem.

   e. SSH into one of the nodes in the management subsystem and become root with `sudo -i`.
   f. Create the cronjob resource in the cluster by running the following command:

```
kubectl apply -f /home/apicadm/configsync-cronjob-ova.yaml
```

The API Connect operations tool: apicops

Use the apicops operations tool to check the health, troubleshoot, and run maintenance operations on your API Connect installation.

The API Connect apicops tool is in active development and is available to download from github.com. It is available for Linux® and macOS. All suggestions, feedback, and bug reports can be raised in this repository.

Installing

Download the binary for your operating system from latest release. Rename the file to **apicops**, and give it execute permission: `chmod +x apicops`.

Running apicops

Apicops requires access to your Kubernetes or OpenShift environment. For more information about using apicops, see: Requirements for running apicops.

Configuring and managing your server environment

You configure and manage the servers that comprise your IBM® API Connect on-premises cloud by using the Cloud Manager user interface.

The Cloud Manager user interface is the part of IBM API Connect that enables a Cloud Administrator to define and manage the API Connect on-premises cloud.

You can use the Cloud Manager to define the API Connect cloud by performing the following tasks:

- Create Provider organizations and invite users to serve as the owner
- Create and manage user roles and role defaults
- Create availability zones for services
- Register the relevant servers that will provide the gateway, analytics, and portal services
- Associate an analytics service with a gateway to enable reports for API Events
- Configure resources for user authentication, TLS security, and OAuth providers and make the resources visible to all or selected provider organizations
- Connect to an existing SMTP mail server and edit templates for system-generated emails
- Set the default gateway service for catalogs

- **Cloud Manager configuration checklist**
  A summary, with links, of the key initial configuration tasks that you must complete in the Cloud Manager user interface after installing and deploying IBM API Connect, and further supplementary tasks.
- **Activating your Cloud Manager user account**
  If you have been invited to be an administrator of your API Connect cloud, you must activate your account by using the activation link that was sent by another administrator. You can then access the Cloud Manager user interface.
- **Accessing the Cloud Manager user interface**
  How to navigate to and login to the Cloud Manager user interface.
- **Defining your Cloud Manager topology**
  To define your API Connect on-premises cloud, you define availability zones and register services within those zones to securely create, promote, and track APIs. A Default Availability Zone is provided on installation.
- **Managing authentication and security**
  Secure Cloud Manager and API Manager as well as your Catalogs with a user registry. Secure your APIs with OAuth. Create TLS profiles to ensure that information you share among web servers will not be stolen or tampered with.
- **Configuring the cloud settings**
  Before you can add a provider organization to your cloud, you must specify your cloud settings to configure an email server for notifications as well as user registry options and catalog defaults.
- **Configuring API governance in the Cloud Manager**
  How to add API governance rulesets in the Cloud Manager, to validate and enforce organizational governance policies and best practices across your API development process.
- **Administering provider organizations**
  Manage the provider organizations who have accounts to publish APIs in your API Connect cloud.
- **Administering members and roles**
  Cloud Administrators can add members and assign them roles to enable them to work in Cloud Manager. The Cloud and Topology Administrators can also create roles and role defaults. You can also delete users to prevent them from accessing Cloud Manager.
- **Monitoring the cloud**
  API Connect generates events to monitor the status of your cloud.
- **Changing your Cloud Manager password and profile information**
  You can change your Cloud Manager password and update your profile information.
- **Resolving login problems by increasing HTTP header size**
  You can resolve login problems for the Cloud Manager UI by increasing the maximum HTTP client header size.
- **Onboarding a new admin for Cloud Pak for Integration**
  Use the API Connect toolkit CLI to create a new admin account for use with Cloud Pak for Integration.
- **Extending the Gateway server behavior**
  To support your enterprise requirements, you can extend the Gateway servers within IBM API Connect to provide extra enforcement behavior.
- **Cloud Manager Tutorials**
  Tutorials for using the Cloud Manager user interface in IBM API Connect.

Cloud Manager configuration checklist

A summary, with links, of the key initial configuration tasks that you must complete in the Cloud Manager user interface after installing and deploying IBM® API Connect, and further supplementary tasks.

Key initial tasks

Task	Description
Log in to the Cloud Manager user interface	A default admin user account is provided for you to begin your configuration tasks.
Configure an email server for notifications	The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service.
Specify your email server as the notification server	Select the email server to be used as the notification server, and configure the sender name and email address to be included in the emails.
Register a Gateway service	A Gateway service represents a cluster of gateway servers that host published APIs and provide the API endpoints used by client applications. Gateways execute API proxy invocations to backend systems and enforce API policies including client identification, security and rate limiting.
Register an Analytics service	The Analytics service collects API event data from the Gateway service.
Associate the Analytics service with a Gateway service	You must associate an Analytics service with each of the Gateway services from which you want to collect API event data.
Register a Portal service	The Portal service provides a developer portal used by application developers to discover APIs and onboard consumers.

Task	Description
Create a provider organization	For developers to be able to publish APIs, and for API managers to be able to manage the API lifecycle, they must be a member of a provider organization. You must create a least one provider organization specifying the owner, who can add further members.

Further supplementary tasks

Task	Description
Configure authentication and security	API Connect provides default local user registries, but you can configure your own user registries, of various types, for user interface and API access control. You can also configure TLS profiles for securing data transmission over HTTPS, and OAuth providers for securing third-party website or application access to APIs.
Create an Availability Zone	An Availability Zone is a logical grouping of one or more API Connect services. A default Availability Zone is provided on installation, but you can add further Availability Zones to reflect your data center topology.
Administer members and roles	You can add further users as members of the administration organization, and control their permissions by assigning predefined roles. You can also create your own custom roles.
Monitor your API Connect cloud	API Connect generates events to allow you to monitor the status of your cloud.

Related concepts

- Cloud Manager Tutorials

Activating your Cloud Manager user account

If you have been invited to be an administrator of your API Connect cloud, you must activate your account by using the activation link that was sent by another administrator. You can then access the Cloud Manager user interface.

Before you begin

A cloud administrator invited you to join API Connect .

Procedure

1. Complete the following steps to activate your Cloud Manager account:

   If the Identity Provider uses LDAP
   : An invitation email with an activation link is sent. Click the activation link, or paste it in a browser, to log in directly with your LDAP user credentials. Upon authentication, the API Connect user record is updated from the backend identity provider.

   If the Identity Provider uses a local registry
   : An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your first name, last name and password. Passwords must have a minimum of 8 characters and contain characters from at least three of the four following categories:
     - Uppercase letters
     - Lowercase letters
     - Numbers
     - Special characters (for example: ! # $ %)

     Note:
     - The email address that you enter on the sign up page must match the email address to which the invitation email was sent, otherwise the account activation fails.
     - If you previously had an account that was removed, and you are being invited again, you must re-activate your account by using the Sign In option on the page, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail.

   If the Identity Provider uses an authentication URL
   : An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your credentials, which then become your user name and password. Upon authentication, the API Connect user record is updated from the backend identity provider.

   If multiple user registries are available for selection on the sign up page, then make sure that the correct registry for your Cloud Manager account is selected. You might need to ask your administrator which user registry is appropriate for your account.

   For information on configuring user registries and making them available for Cloud Manager login, see User registries overview and Selecting user registries for Cloud Manager and API Manager.

2. Click Sign up to complete your registration, then click Sign in to open the Cloud Manager login page.

Results

You have created your API Connect administrator account.

What to do next

Log in to the Cloud Manager user interface and, depending on your role, start to manage the cloud topology, and work with provider organizations and cloud resources.

To access the Cloud Manager login page in the future, use the following URL:

`https://host/admin`

where *host* is the fully qualified host name or IP address of the Management server.

Related concepts

- Managing authentication and security
- Configuring the cloud settings

Related tasks

- Accessing the Cloud Manager user interface
- Defining your Cloud Manager topology
- Administering provider organizations
- Administering members and roles

Accessing the Cloud Manager user interface

How to navigate to and login to the Cloud Manager user interface.

Procedure

1. Open a browser and enter the URL for the cloud-admin-ui endpoint followed by /admin. Note that this is a secure connection that uses HTTPs, for example: **`https://<cloud-admin-ui>/admin`**.
   - VMware installations: The cloud-admin-ui endpoint was set during installation of the Management subsystem by the following command: **`apicup subsys set mgmt cloud-admin-ui <endpoint>`**. Append **`/admin`** to this value.
   - Kubernetes, OpenShift, Cloud Pak for Integration installations: The cloud-admin-ui endpoint is specified in the Management subsystem CR, in the **`Endpoints`** section, for example:

     ```
     Endpoints:
       ...
       Name:   admin
       Type:   UI
       Uri:    https://example.com/admin
       ...
     ```

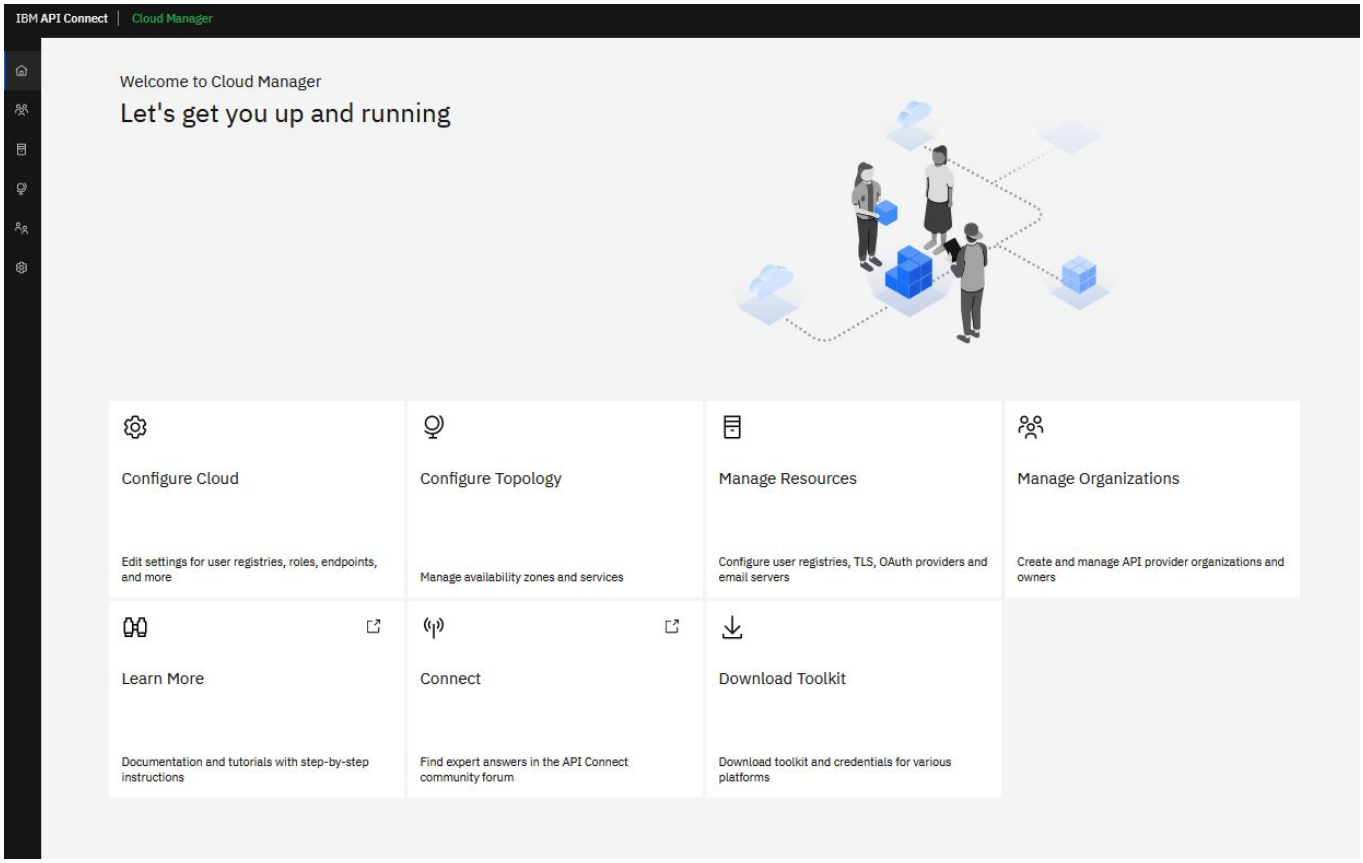
2. Enter the Cloud Administrator user name and password.
   - Kubernetes, VMware, and OpenShift individual Management subsystem install:
     If you are logging in for the first time, enter `admin` for the user name and `7iron-hide` for the password. Otherwise, enter `admin` for the user name and your own password.
   - OpenShift top-level CR install:
     The default user name is `admin`. If you are logging in for the first time, run the following commands to retrieve the initial password, which is generated automatically during deployment:

     ```
     oc get secret -n <namespace> | grep mgmt-admin-pass

     oc get secret -n <namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -d && echo ""
     ```

3. If your cloud contains multiple user registries, as configured on the Settings > User Registries screen, then choose the user registry that contains your login credentials. You might need to ask your administrator which user registry is appropriate for your account.
4. Click Sign In.
   Important: The first time that you access the cloud console, you must, for security reasons, enter a new password and your email address. If you forget your password and request a password reset, the notification email is sent to this email address. The email is sent by the email server that is configured in the **Notifications** section of the cloud **Settings**.

Results

The Cloud Manager user interface opens to the home page, as shown in the following screen capture:



What to do next

Configure the email server so you can reset your password if needed and send other notifications. See Tutorial: Configuring the Cloud.

Configure your topology by registering services. See Tutorial: Configuring the Cloud.

Create a Provider Organization. See Tutorial: Creating a Provider Organization.

Related tasks

- Changing your Cloud Manager password and profile information

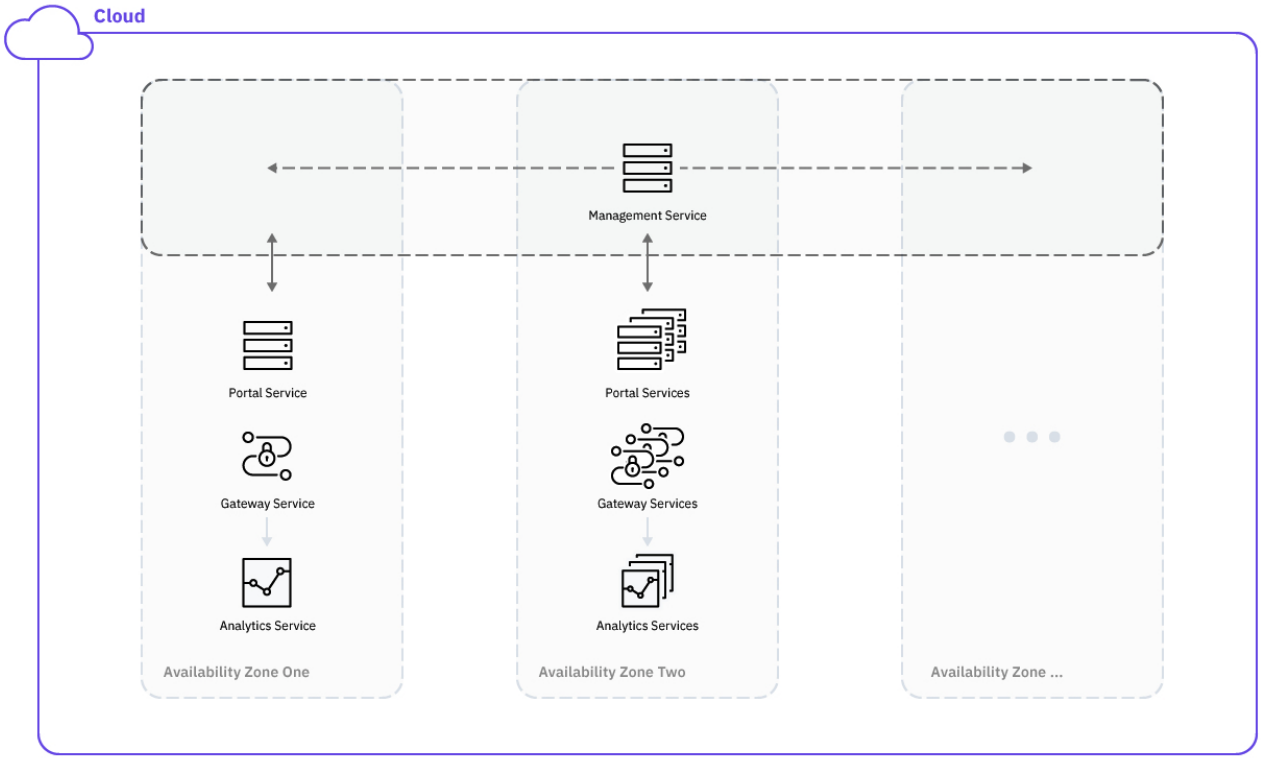
Defining your Cloud Manager topology

To define your API Connect on-premises cloud, you define availability zones and register services within those zones to securely create, promote, and track APIs. A Default Availability Zone is provided on installation.

About this task

The Cloud Manager topology consists of Availability Zones that contain the API Connect services. Availability Zones can contain one or more gateway services, analytics services, and portal services.

The following diagram illustrates the topology for services and Availability Zones in an API Connect network:



One of the following roles is required to register and manage services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Following are the tasks involved in defining the API Connect topology:

- Create one or more Availability Zones.
- Register one or more Gateway, Analytics, and Portal services in each Availability Zone
- Associate an Analytics service with each Gateway service
- Set the visibility for the services

Once the cloud is defined and an email server has been configured, you invite other users to access API Manager to create APIs. You expose these APIs to the development community through the Developer Portal user interface.

To define the API Connect cloud, complete the following tasks:

- **Creating an availability zone**
  An Availability Zone is a logical group of one or more API Connect services. The Default Availability Zone is created during installation.
- **Registering a gateway service**
  A gateway service is required to handle incoming traffic for APIs.
- **Registering an analytics service**
  Configure at least one analytics service in your API Connect on-premises cloud. The analytics service is always associated with a gateway service, from which it collects API event data.
- **Registering a portal service**
  Define one or more Portal services in your API Connect on-premises cloud.
- **Associating an analytics service with a gateway service**
  To collect data about your API usage and other statistics, you must associate an analytics service with each gateway service. You can associate multiple gateway services with one analytics service, but each gateway can be associated with only one analytics service.
- **Setting visibility for a service**
  The visibility setting determines which provider organizations can access a service.

Related concepts

- Configuring and managing your server environment

Creating an availability zone

An Availability Zone is a logical group of one or more API Connect services. The Default Availability Zone is created during installation.

About this task

Availability zones allow you to logically group API Connect services to suit your environment. For example, you can group gateway services according to the region or data center they are located in.

The Default Availability Zone is created during the installation process.

When you create an availability zone, you should provide a descriptive title; a name is generated automatically for internal identification. For example:

Table 1. Example availability zones

Title	Name
US Availability Zone	us-availability-zone
Hampshire Availability Zone	hampshire-availability-zone
Newport Availability Zone	newport-availability-zone
London South Availability Zone	london-south-availability-zone

One of the following roles is required to add and manage Availability Zones:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Procedure

Follow these steps to add one or more additional Availability Zones to your on-premises cloud:

1. In the Cloud Manager, click Topology.
2. You will see the current Availability Zones configured in your cloud. The Default Availability Zone is created at installation. To add another Availability Zone, choose Create Availability Zone.
3. Enter the following values:

Field	Description
Title (required)	Enter a descriptive title for the availability zone. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.

4. Click Create to complete the operation.

Results

The availability zone will be added on the Topology page. You can now add gateway, analytics, and portal services to the availability zone.

Related tasks

- Registering a gateway service
- Registering an analytics service
- Registering a portal service

Registering a gateway service

A gateway service is required to handle incoming traffic for APIs.

Before you begin

Before registering a Gateway service in Cloud Manager, the DataPower® API Connect Gateway Service has to either be installed as a subsystem in your Kubernetes cluster or enabled on the DataPower appliance. For a Kubernetes environment, see DataPower Gateway subsystem on Kubernetes. For appliances, see *Configuring the API Connect Gateway Service* in the appropriate version of the DataPower documentation.

About this task

A Gateway service represents a cluster of gateway servers that host published APIs and provide the API endpoints used by client applications. Gateways execute API proxy invocations to backend systems and enforce API policies including client identification, security and rate limiting.

One of the following roles is required to register and manage a gateway services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Note: You can also register, and manage, gateway services by using the developer toolkit CLI; for details, see apic gateway-services.

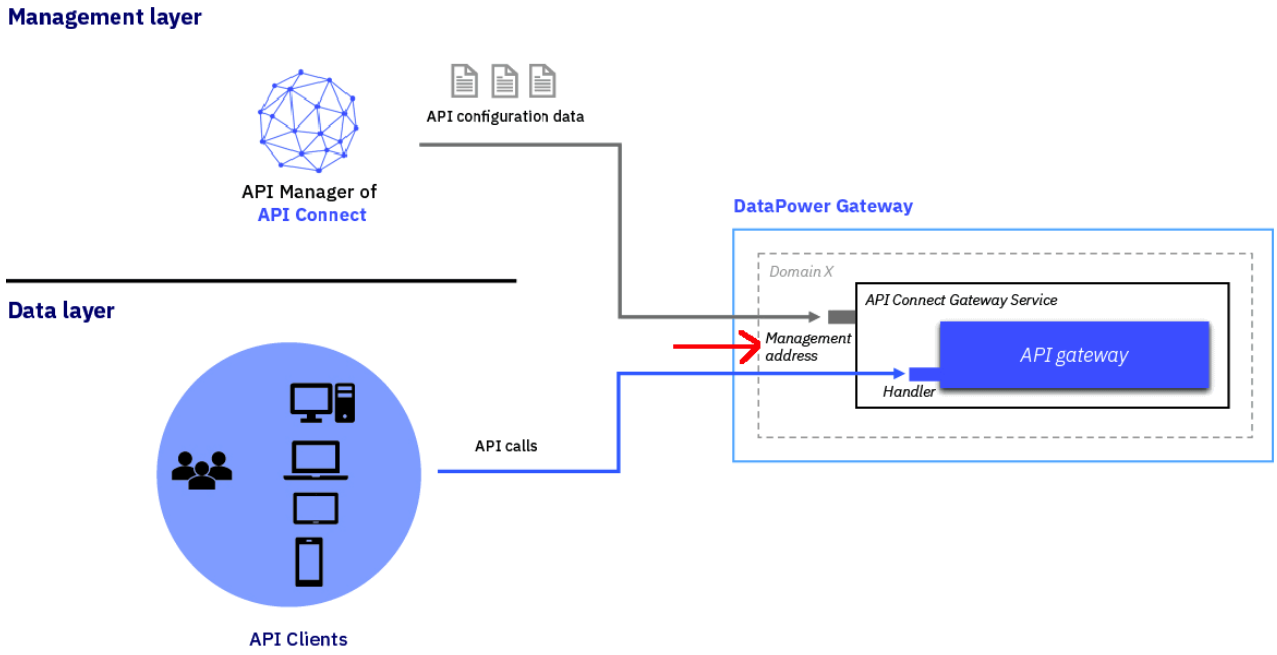
Procedure

Complete the following steps to configure a Gateway service for your cloud:

1. In the Cloud Manager, click Topology.
2. From the Availability Zone that will contain the Gateway service, select Register Service.
3. On the Configure Service page, select DataPower Gateway as the service type. Select the Gateway type that you want to create, either DataPower Gateway (v5 compatible) or DataPower API Gateway. For a description of the gateway types, see API Connect gateway types
4. Enter the values to configure the Gateway service. You will need to obtain the endpoints from your deployment configuration. For a Kubernetes environment, the endpoints are configured by the following values in the CR (custom resource) file. For an appliance, the endpoint is configured in DataPower.

Field	Description
Title (required)	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.
Management endpoint on the gateway service: Endpoint (required)	Enter the API Connect Gateway Service endpoint. • For a Kubernetes environment, the Management Endpoint is the endpoint created in the CR (custom resource). See DataPower Gateway subsystem on Kubernetes for more information. • For an appliance, the Management Endpoint is the *Management address* to the API Connect Gateway Service shown in the gateway service connection diagram. For one gateway, this takes the form *https://<ip-address-for-gateway>:3000*. For multiple gateways, it would be the address:port of the load balancer • If you are using **`in-cluster`** communication, set this to the service endpoint URL for the gateway, which has the format: **`https://<name>.<namespace>.svc`** where: ◦ *<name>* is the **`name`** property from the gateway subsystem CR. If you are using the top-level CR this is **`<apic instance name>-gw`** ◦ *<namespace>* is the namespace that the gateway CR exists in.
Management endpoint on the gateway service: TLS client profile (required)	Specify the TLS client profile to use when the management subsystem initiates communication with the gateway. The client profile contains the TLS client certificate that is presented to the gateway when establishing a secure connection. Important: • Do not leave this property unset. You must specify a client profile for secure management to gateway communication (mTLS). If you do not want to use mTLS, then you can configure JWT instead: ◦ Use JWT instead of mTLS on VMware. ◦ Use JWT instead of mTLS on Kubernetes and OpenShift. • The default profile to use is Gateway management client TLS client profile:1.0.0. On a Kubernetes or OpenShift deployment where the management and gateway subsystems are in the same namespace, the gateway service is automatically configured (during installation) to trust the Gateway management client TLS client profile:1.0.0. • If you want to use a different client certificate, do not update the Gateway management client TLS client profile:1.0.0, create a new client profile that uses the certificate you want. See Creating a TLS Client Profile. • If you are registering a gateway appliance, then configure your gateway to trust the client profile that you select. For more information about configuring TLS on your gateway appliance, see ◦ Configuring DataPower API Gateway. ◦ Configuring DataPower Gateway (v5 compatible). Do not configure your gateway appliances with client authentication disabled.
Management endpoint on the gateway service: Use **`in-cluster`** communication switch	If you want to use **`in-cluster`** communication between the management and gateway subsystems, then enable this switch. For more information about this option, see In-cluster or external communication between subsystems. If you are not sure, then leave this switch disabled (the default).
Management endpoint on the gateway service: Use JWT for gateway authentication to analytics service	If you plan to associate an analytics service with this gateway, and you want to use JWT authentication between the gateway and analytics service, then enable this switch. If this switch is not enabled, then mutual TLS authentication is used between the gateway and analytics service. For more information about JWT security, see Enable JWT security instead of mTLS
API invocation endpoint: API endpoint base (required)	Enter the base portion of the URL that maps to the base portion of the URL for incoming API traffic. It is a public FQDN with additional paths that are specific to your API calls. For example: **https://api.mycompany.com** Note: If you are using the DataPower API Gateway, HTTP/2 is enabled automatically on the API invocation endpoint. If you are using the DataPower Gateway (v5 compatible), HTTP/2 is not supported on the API invocation endpoint.
API invocation endpoint: Server Name Indication (SNI) - Host name	For supporting Server Name Indication (SNI) at the API Endpoint Base. The default hostname of '*' is required to allow all hosts. Enter other host names as needed. Wild card format is supported. The SNI capability enables you to serve multiple TLS secure host names through the same Gateway service, using the same IP address and port, without requiring them to use the same TLS profile. Note: To allow requests from clients that don't support SNI, you **must** include a host name value of '*'.
API invocation endpoint: Server Name Indication (SNI) - TLS server profile	The TLS server profile that supports the given hostname for SNI. The server profile that is selected here contains the server certificate that is presented to callers of the APIs you publish on the gateway. By default this profile is set to Default TLS server profile. Best practice is to create your own TLS server profile that contains the certificate you want on your API invocation endpoint. For more information about configuring TLS server profiles, see TLS profiles overview. Restriction: If you are using the DataPower API Gateway, HTTP/2 is enabled automatically on the API invocation endpoint. Therefore, using a TLS server profile that supports only TLS protocol version 1.3 is not supported and will result in the gateway being unable to receive API traffic. You must use a TLS server profile that includes TLS protocol version 1.2 as one of its supported protocols. This restriction does not apply if you are using the DataPower Gateway (v5 compatible), which does not support HTTP/2.
OAuth shared secret (optional)	For sites using native OAuth providers, enter the shared secret that will be used by all API calls going through the gateway. Note: The specified shared secret must be 64 characters (64 bytes) in length, prefixed with **`0x`**, and must consist only of hexadecimal characters. For example: **`0xa354282f227c10250511ae9c9e8c7ed9f4f1bd0d7c04cb6d5bd178f8c62296e3`**

The following diagram illustrates the gateway service connection:



5. When you are finished, click Save.

Results

The Gateway service is added to the appropriate Availability Zone for your cloud.

What to do next

Add additional gateway services. Add one or more analytics services. Add one or more portal services. Associate the gateway service with an analytics service. Set the visibility for the gateway.

Related tasks

- Registering a portal service
- Registering an analytics service
- Associating an analytics service with a gateway service
- Setting visibility for a service

Registering an analytics service

Configure at least one analytics service in your API Connect on-premises cloud. The analytics service is always associated with a gateway service, from which it collects API event data.

Before you begin

You must complete the following tasks:

- Creating an availability zone
- Registering a gateway service

About this task

The analytics service collects API event data from the gateway service. In Cloud Manager, you configure the analytics service and then associate it with one or more gateways. Multiple gateways can be associated with a single analytics service, but each gateway can be associated with only one analytics service.
One of the following roles is required to register and manage analytics services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

When you register the analytics service, you must select from one of three inter-subsystem communication methods. Before you decide which method to use, familiarize yourself with the difference between **`external`** and **`in-cluster`** communication: In-cluster or external communication between subsystems. If you are an OVA user, or are in doubt as to what communication method to use, then use the default **`external`** setting.

The analytics subsystem has two separate communication flows:

- Management to analytics. The management service communicates with the analytics subsystem to query for analytics data to present in the UI dashboards. This flow is known as "analytics queries".
- Gateway to analytics. The gateway sends API event data to the analytics service. This flow is known as "analytics ingestion".

What you select depends on what is most suitable for your network configuration. The options are:

Use `external` communication for both ingestion and queries
: This option is the default, and is what is used for analytics services that are upgraded from a pre-10.0.5.3 release. For OVA deployments, or if your analytics subsystem is in a different environment then `external` is the only option that you can use. Both the gateway and the management service communicate with the analytics service by using the external endpoint, which is a Kubernetes `ingress` or OpenShift `route` depending on your platform.

Use `in-cluster` communication for both ingestion and queries
: You can select `in-cluster` if you have a Kubernetes or OpenShift deployment where all subsystems are within the same cluster. When this option is selected, the management, gateway, and analytics subsystems communicate with each other through internal service endpoints rather than externally accessible `ingresses` (Kubernetes) or `routes` (OpenShift).

Use `in-cluster` for ingestion and `external` for queries, or vice versa
: You can select this option if you have a Kubernetes or OpenShift deployment and you have some of your subsystems installed in the same cluster. When this option is selected, you can use different communication methods for the different analytics communication flows. For example, the management to analytics communication can use `in-cluster`, and the gateway to analytics can use `external`. You might choose this configuration if your gateway is in a different cluster to the rest of your subsystems.

Procedure

Complete the following steps to configure the analytics services for your cloud:

1. In the Cloud Manager, click Topology.
2. Select Register Services > Analytics in the Availability Zone where you have the gateways you want to associate with the analytics service.
3. Enter the values to configure the analytics service:

Field	Description
Title	Enter a descriptive title for the analytics service.
Name	This field is auto-populated by the system and used as the internal field name.
Summary	Enter a brief description.
Service endpoint configuration: Communication method.	Select your communication method, for more information on the options see, Communication types. If your API Connect deployment is OVA then you must select the first option: Use `external` communication for both ingestion and queries.

Service endpoint configuration: `External` analytics endpoint.	Enter the fully qualified domain name for the analytics ingestion endpoint that you defined during installation. To determine the endpoint, run the following command:
	• Kubernetes
	`kubectl get ingress -n <namespace>`
	The endpoint has `-ai-` in its name:
	`NAME CLASS HOSTS ADDRESS PORTS AGE`
	`def-analytics-ai-endpoint nginx a7s-in.example.com 80, 443 87d`
	• OpenShift
	`oc get routes -n <namespace>`
	The endpoint has `-ai-` in its name:
	`NAME CLASS HOSTS ADDRESS PORTS AGE`
	`def-analytics-ai-endpoint nginx a7s-in.example.com 80, 443 87d`
	• VMware
	`apicup subsys get <analytics subsystem>`
	The endpoint is the value of property `analytics-ingestion`.
	This field is disabled if "Use `in-cluster` communication for both ingestion and queries" is set as the communication type.

Service endpoint configuration: TLS client profile for `external` analytics endpoint.	Select the TLS Client Profile to use for management to analytics subsystem communication. The Client Profile applies to the endpoint that you specified in the previous row. The default is "Analytics Ingestion TLS client profile". This field is disabled if "Use `in-cluster` communication for both ingestion and queries" is set as the communication type.
Service endpoint configuration: `In-cluster` analytics endpoint.	Enter the fully qualified domain name for the analytics ingestion service endpoint. The format of the service endpoint is `https://<name>.<namespace>.svc` where: • *<name>* is the `name` property from the analytics subsystem CR. If you are using the top-level CR this is `<apic instance name>-a7s`. • *<namespace>* is the namespace that the analytics CR exists in. This field is disabled if Use `external` communication for both ingestion and queries is set as the communication type.

Field	Description
Service endpoint configuration: TLS client profile for `in-cluster` endpoint.	Select the TLS Client Profile to use for management to analytics subsystem communication. The Client Profile applies to the Endpoint that you specified in the previous row. The default is "Analytics Ingestion TLS client profile". This field is disabled if Use `external` communication for both ingestion and queries is set as the communication type.
Service endpoint configuration: Use `in-cluster` endpoint for analytics queries.	If you want the management subsystem to use the `in-cluster` service endpoint to communicate with the analytics subsystem, then select this option. If this option is not selected, then the management subsystem uses the `external` endpoint. Selection of `external` or `in-cluster` communication between the gateway and analytics subsystem is done when the gateway is associated with the analytics service, see Associating an analytics service with a gateway service. This setting is enabled when "Use `in-cluster` for ingestion and `external` for queries, or vice versa" is selected.

4. Click Save to complete the operation.

Results

The analytics service is added to your cloud settings.

What to do next

- Associate the analytics service with one or more gateway services: Associating an analytics service with a gateway service.
- Set preferred retention and rollover settings for the new analytics service: Retention and rollover.

Related tasks

- Registering a gateway service
- Registering a portal service

Registering a portal service

Define one or more Portal services in your API Connect on-premises cloud.

Before you begin

You must complete the following tasks:

- Creating an availability zone
- Registering a gateway service
- Configuring an email server for notifications
- Setting up notifications

Note:
When you create or register a Developer Portal service, the Portal subsystem checks that the Portal web endpoint is accessible. However sometimes, for example due to the complexity of public and private networks, the endpoint cannot be reached. The following example shows the errors that you might see in the `portal-www` pod, admin container logs, if the endpoint cannot be reached:

```
An error occurred contacting the provided portal web endpoint: example.com
The provided Portal web endpoint example.com returned HTTP status code 504
```

In this instance, you can disable the Portal web endpoint check so that the Developer Portal service can be created successfully.
To disable the endpoint check, complete one of the following updates depending on your platform:

On Kubernetes, OpenShift, and IBM® Cloud Pak for Integration

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

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

On VMware

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

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

Run the following commands:

```
apicup subsys set <ptl_subsys> extra-values-file <path-to-ptl-extra-values-yaml-file>

apicup subsys install <ptl_subsys>
```

About this task

Each Availability Zone contains one or more Portal services. The Portal service provides a Developer Portal used by application developers to discover APIs and onboard consumers. An email server must be configured and set as the email server for the cloud before registering a Portal service.
One of the following roles is required to register and manage Portal services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Important: It's not recommended to have more than 100 sites per Developer Portal service. Note that it's not necessary to have a Portal site for every Catalog, for example Catalogs that are only for API Developers don't need a Portal site, as the APIs can be tested by using credentials from the API Manager. If more than 100 sites are required, you should configure additional Developer Portal services.

Procedure

Complete the following steps to configure the Portal services for your cloud:

1. In the Cloud Manager, click Topology.
2. In the Availability Zone that will contain the Portal service, select Register Services > Portal.
3. Enter the values to configure the Portal service.

Field	Description
Title (required)	Enter a descriptive title for the portal service. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Summary (optional)	Enter a brief description.
Management endpoint on the portal service: Endpoint (required)	Enter the IP address, fully-qualified host name, service, or ingress name. Used for communication with API Manager. • In a Kubernetes deployment, the endpoint is defined in the CR by `api.portal.$STACK_HOST` . See Developer Portal subsystem on Kubernetes. • If configured during installation in a VMware environment, it is the `apicup subsys set portal portal-admin` value. • If you are using `in-cluster` communication, set this to the service endpoint URL for the portal, which has the format: `https://<name>.<namespace>.svc` where: ◦ *<name>* is the `name` property from the portal subsystem CR. If you are using the top-level CR this is `<apic instance name>-ptl` ◦ *<namespace>* is the namespace that the portal CR exists in.
Management endpoint on the portal service: TLS client profile (optional)	Select the TLS Client Profile that will be used to communicate with the portal service. The profile applies to the Management Endpoint. Note: TLS protocol version 1.3 is not supported.
Management endpoint on the portal service: Use `in-cluster` communication switch	If you want to use `in-cluster` communication between the management and portal subsystems, then enable this switch. For more information about this option, see In-cluster or external communication between subsystems. If you are not sure, then leave this switch disabled (the default).
Portal website URL (required)	The URL that will be used for public access to the portal. • In a Kubernetes deployment, the endpoint is defined in the CR by `portal.$STACK_HOST` . See Developer Portal subsystem on Kubernetes. • If configured during installation in a VMware environment, it is the `apicup subsys set portal portal-www <portal>.<hostname>.<domainname>` value. Multiple portal-www endpoints may be configured, as described here in these topics: Defining multiple portal endpoints for a Kubernetes environment and Defining multiple portal endpoints for a VMware environment.

4. When you are finished, click Save.

Results

The Portal service is configured in your cloud and can be used to publish APIs.

Related tasks

- Registering a gateway service
- Registering an analytics service

Associating an analytics service with a gateway service

To collect data about your API usage and other statistics, you must associate an analytics service with each gateway service. You can associate multiple gateway services with one analytics service, but each gateway can be associated with only one analytics service.

About this task

The associated analytics service receives API event data from the gateway service.

To associate an analytics service with a gateway, you must first configure at least one analytics service and one gateway service.

One of the following roles is required to associate an analytics service with a gateway service:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Follow these steps to associate an analytics service with a gateway service:

Procedure

1. In the Cloud Manager, click Topology.
2. In the section for the Availability Zone you want to work on, locate the gateway service that requires an analytics service in the SERVICE column.
3. Locate the ASSOCIATED ANALYTICS SERVICE column.
4. If an analytics service is configured, the Associate Analytics service link is enabled for each gateway service. Click Associate Analytics Service in the same row as the gateway service that requires an analytics service.
5. In Associate Analytics Service, you see the name of the gateway service and a list of analytics services that have been configured in your cloud.
6. Select the analytics service to associate with the gateway service by adding a checkmark next to the analytics services.
7. If you specified `in-cluster` communication when you registered the analytics service, you can enable the gateway to send API event data to the internal analytics endpoint. Enable the "Use internal endpoint for ingestion of analytics data from the gateway service" switch. For more information about `in-cluster` communication, see Communication types.
8. Click Associate.
9. To remove the associated analytics service, select Unassociate analytics service from the actions menu. The gateway stops sending API event data to the analytics service.

Results

The analytics service is associated with a gateway service and starts to receive API event data from that gateway.

Related tasks

- Registering a gateway service
- Registering an analytics service

Related information

- Installing API Connect into a Kubernetes environment
- Deploying the Management OVA file

Setting visibility for a service

The visibility setting determines which provider organizations can access a service.

Before you begin

You must complete the following tasks:

- Create additional availability zones if needed, see Creating an availability zone
- Register at least one gateway service, see Registering a gateway service
- Register other services.

About this task

The visibility setting controls which provider organizations can use a service. The default visibility setting is Public.

One of the following roles is required to set the visibility for services:

- Administrator
- Topology Administrator
- Owner
- A custom role with the `Topology:Manage` permission

Procedure

Follow these steps to set the visibility for the services in your on-premises cloud:

1. In the Cloud Manager, click Topology.
2. From the list of Services, choose Set visibility from the actions menu next to the name of the service that requires the visibility setting.
3. Select the visibility setting for the service. The options are:
    - Private - the service is not visible and cannot be used by any provider organization

- Public - the service is visible and can be used by all provider organizations
- Custom - the service is visible only to the provider organizations designated by you

4. For Custom visibility, select the provider organizations that will be able to use the service.
5. Click Make visible to complete the operation.

Results

For Private, the service cannot be used by any provider organizations. For Public, the service can be used by all provider organizations. For Custom, the service can be used by the provider organizations that you designate.

Managing authentication and security

Secure Cloud Manager and API Manager as well as your Catalogs with a user registry. Secure your APIs with OAuth. Create TLS profiles to ensure that information you share among web servers will not be stolen or tampered with.

As Cloud Administrator or Topology Administrator (or with a custom role that contains the `Settings:Manage` permission), you can configure the following authentication and security mechanisms:

- User registries to authenticate users of Cloud Manager and API Manager and of your Catalogs and APIs.
- OAuth providers to provide protection for APIs.
- TLS profiles to secure transmission of data through the Gateway to external web sites and among web servers.

The following topics describe how to configure user registries, OAuth providers, and TLS profiles:

- **User registries overview**
  API Connect supports several types of user registries for authenticating users. The credentials for all users of Cloud Manager, API Manager, and the Developer Portal must be stored in a user registry.
- **TLS profiles overview**
  API Connect supports TLS Profiles for securing data transmission over HTTPS.
- **OAuth Provider overview**
  API Connect supports OAuth Specification 2.0, for both Native and Third party implementations.

Related concepts

- User registries overview
- OAuth Provider overview
- TLS profiles overview

User registries overview

API Connect supports several types of user registries for authenticating users. The credentials for all users of Cloud Manager, API Manager, and the Developer Portal must be stored in a user registry.

Introduction to user registries

A user registry holds unique user account credentials, primarily usernames and passwords, which are accessed during authentication for logging into Cloud Manager, API Manager, the Developer Portal, and also for calling APIs (if the API is configured to use Basic Authentication in the Security Definition). The Cloud Administrator configures user registries to ensure that all users are successful at logging in.

User registries also provide security when calling APIs. For information on configuring Security Definitions for APIs, see Creating a basic authentication security definition.

User registries are configured as Resources. Once the user registry has been configured as a resource in Resources > User registries, the next step is indicate the active user registries for your cloud in Settings > User registries. The user registry must be made active in your cloud in order to make it available for authentication. When a user logs into Cloud Manager or API Manager, the specified user registry is queried for credentials to confirm the user's identity. To make a user registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Settings > Onboarding; for more information see Creating and configuring Catalogs.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access. In Cloud Manager and API Manager, a registry cannot be changed after a user is invited to be the owner of a provider organization, even if the invitation is not yet accepted.

User registries in API Connect serve the following primary functions:

- To authenticate a user at login time based upon username and password.
- To store basic profile information such as first name, last name, and email address.
- To provide secure access to Catalogs.
- To provide Basic Authentication for APIs when called by an application.

In order to log in to Cloud Manager, API Manager, the Developer Portal, or access a catalog, a user must have valid credentials (username and password) stored in a user registry that is configured in API Connect.

The Security Definition for an API can be configured to require a username and password, called the Basic Authentication method. With Basic Authentication, the username and password are included in the HTTP authorization header, and the credentials are verified through user registry.

User registries that are configured in Cloud Manager have the following characteristics:

- They are available to Cloud Manager, API Manager, the Developer Portal, and for Basic Authentication for APIs.
- They are available to provider organizations, as determined by the visibility setting.
- They can be edited and deleted only in Cloud Manager.

User registries that are configured in API Manager have the following characteristics:

- They are available to the Developer Portal (for Catalogs) and for Basic Authentication for APIs.
- They are available only to the Provider Organization that created them.

User registries supported by API Connect

API Connect integrates with several types of user registries to accommodate all security solutions. You can use your corporate LDAP registry, an Authentication URL, or an LUR to provide secure access to Cloud Manager, API Manager, the Developer Portal, and APIs. API Connect includes two internal databases that serve as local user registries, or LURs. The Providers LUR supports credentials for Provider organizations and the Admin LUR supports credentials for the Administrator organization. Multiple registries may be configured.

The following user registry types can serve as a resource in API Connect:

- Local user registry (LUR) - An internal database of usernames and passwords stored on the local server. Contains the Admin user account which may not be modified or deleted.
- LDAP - A user registry definition that points to an existing corporate LDAP directory. May be set up as case-sensitive.
- Authentication URL - Accesses a URL that points to a service for validating user credentials.
- OpenID Connect (OIDC).

- **Configuring an Authentication URL user registry**
  An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.
- **Configuring a shared custom user registry**
  You can configure a custom user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal.
- **Configuring an LDAP user registry in the Cloud Manager**
  You can use the Cloud Manager UI to configure an LDAP user registry as a shared resource to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.
- **Configuring a Local User Registry**
  A Local User Registry (LUR) can be configured to provide user authentication for both Cloud Manager and API Manager.
- **Configuring an OIDC user registry in Cloud Manager**
  Configure a shared OIDC user registry for user onboarding and authentication when multi-factor authentication (MFA) is required.
- **Setting visibility for a user registry**
  The visibility setting determines which provider organizations can access a resource, for example, user registries.
- **Deleting a user registry**
  As the Cloud Manager administrator, you can delete user registries.
- **Removing a user from a user registry**
  As an API Connect user, you can remove yourself from an API Connect user registry. As an administrative user, you can remove other users from an API Connect user registry.

Related tasks

- Configuring an Authentication URL user registry
- Configuring a Local User Registry
- Configuring an LDAP user registry in the Cloud Manager
- Setting visibility for a user registry

Configuring an Authentication URL user registry

An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.

About this task

This topic describes how to configure a new Authentication URL user registry as a Resource in your cloud. After the user registry is configured, you must select it for use in your cloud in Settings > User Registries. See Selecting user registries for Cloud Manager and API Manager.

One of the following roles is required to configure user registries.:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note:
API Connect issues an HTTP **`GET`** call to the Authentication URL endpoint, sending the user's credential. The following example shows a call made to an Authentication URL identity provider with an endpoint defined as **`https://myauthurl.example.com/user/authenticate`**:

```
GET /user/authenticate HTTP/1.1
 Host: myauthurl.example.com
 Authorization: Basic c3Bvb246Zm9yaw=
```

If the Authentication URL endpoint returns an HTTP status code of **`200`**, the user authenticates successfully. An HTTP status code other than **`200`** indicates a failed login attempt. API Connect forwards any HTTP Header starting with **`X-`** (with the exception of **`X-Client-Certificate`**), and Cookie to the Authentication URL identity

provider, to aid the authentication decision; for example:

```
GET /user/authenticate HTTP/1.1
 Host: myauthurl.example.com
 Authorization: Basic c3Bvb246Zm9yaw=
 X-Forwarded-For: 8.8.9.9
 X-Custom-Header-From-Customer: special
 Cookie: MyCookie=VGhpc0lzV2lja2VkQW1hemluZw==
```

When a user is presented with the form for completing their API Connect user registration, which fields are pre-populated depends on which fields are returned in the response from the Authentication URL identity provider. If any of the following fields are returned, they will be pre-populated in the registration form:

- `username`
- `email`
- `first_name`
- `last_name`

If the `username` field is not returned, the registration form displays the user name that was provided by the user. The pre-population capability requires that the response from the Authentication URL identity provider satisfies the following conditions:

- The `Content-Type` must be `application/json`.
- The response body format must be JSON.

A sample response is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "username":"myuser",
  "email":"myuser@example.com",
  "first_name":"My",
  "last_name":"User"
}
```

Procedure

1. In the Cloud Manager, click .
2. Select User Registries to see the list of current user registries in your cloud.
3. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
4. Select Authentication URL User Registry and enter the following parameters:

Field	Description
Title (required)	Enter a descriptive name to use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Summary (optional)	Enter a brief description.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the `Title` of the User Registries when rendering them at the login page, rather than the `Display Name`.
URL (required)	Enter the URL for the authentication service. When establishing authentication, API Connect makes a GET call to the URL. The call includes the user name and password it has collected from the user in its authorization header. Either `200 OK` or `401 Unauthorized` will be returned.
TLS Client Profile (optional)	Select a TLS Client Profile to allow secure authentication with a specific web server.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend Authentication URL server: • **Only** select Case sensitive if your backend server supports case-sensitivity. • Do **not** select Case sensitive if your backend server does not support case-sensitivity. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.

Results

The Authentication URL user registry is saved and is available to be selected as a user registry for Cloud Manager or API Manager by selecting it in Settings > User Registries. It can also be used for Basic Authentication in the Security Definition for an API.

Related concepts

- Managing authentication and security
- User registries overview

Related tasks

- Configuring a Local User Registry
- Configuring an LDAP user registry in the Cloud Manager
- Setting visibility for a user registry

Configuring a shared custom user registry

You can configure a custom user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal.

Before you begin

To configure a custom user registry as a resource in the Cloud Manager, the external user directory must be created and available to use with your API Connect ecosystem.

Custom user registries can be used for authenticating users to the Cloud Manager, the API Manager, and the Developer Portal, but cannot be used to secure APIs.

One of the following roles is required to configure a custom user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create a custom user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific custom user registry can be used for authenticating Developer Portal users, while a shared custom user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure a shared custom user registry. If you want to create an organization-specific registry, see Creating an organization-specific custom user registry for more information. Note also that the visibility of a user registry is set to **`shared`** by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see Setting visibility for a user registry.
Note:

- You can also create custom user registries by using the API Connect REST APIs (see the API Connect REST API documentation).

You can use the following instructions to create a writable or a read-only custom user registry.

API Connect provides two methods for creating a shared custom user registry in the Cloud Manager, as described in the following sections:

- Creating a shared custom user registry by using the Cloud Manager UI
- Creating a shared custom user registry by using the developer toolkit CLI

Creating a shared custom user registry by using the Cloud Manager UI

You configure a custom user registry by first creating a custom user registry type that defines your external user registry, and then creating a new custom user registry resource that references the custom user registry type that you just created. To make the registry available for user authentication, you need to add the registry to the Cloud Manager, or to the API Manager, or both, depending on your requirements. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the custom user registry type.

Creating a custom user registry type

Use the following instructions to create a custom user registry type that defines your external user registry.

1. In the Cloud Manager, click Resources.
2. Click Create in the User registry type section.
3. Enter the following information:

Property	Description
Title	A descriptive name of the custom user registry type to display in the UI. For example, *My custom user registry type*.
Name	The name of the user registry type. This name is auto-generated, and is used in CLI commands.
Version	The version number of the custom user registry type.
Summary	A brief description of the custom user registry type.
Properties	Enter the properties of your external user registry. Must include the property: **`remote: true`** This property must be set to **`true`** to indicate that this is an external user registry.

Property	Description
Secured Endpoint	Denotes the external registry secured endpoint section of the custom user registry type.
URL	The secure endpoint of your external user registry interface, where your platform REST API is located. It is made up of the URL and port, for example: `https://custom.ms.com:8888` API Connect will make REST calls to this endpoint for the user registry actions.
TLS Client Profile	Optionally set the TLS Client Profile URL that the custom server requires.
Headers	Optionally complete the user-defined header section to your external registry specification.
Configuration schema	Optionally complete the user-defined configuration schema section to your external registry specification.

4. Click Create. A new user registry with a Type of Custom is displayed in the User registry type list.

Creating a custom user registry resource

Use the following instructions to create a new custom user registry resource that references the custom user registry type that you just created.

1. Click Create in the User Registries section of Resources.
2. Click the Custom user registry tile.
3. Enter the following information:

Property	Description
Custom type	The name of the custom user registry type that you just created. For example, *my-custom-user-registry-type*. Use the drop-down arrow to select different custom user registry types.
Title	A descriptive name to display in the UI.
Name	The name of the custom user registry. This name is auto-generated, and is used in CLI commands.
Summary	A brief description of the custom user registry.
Endpoint	Optional endpoint information.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend server: • **Only** select Case sensitive if your backend server supports case-sensitivity. • Do **not** select Case sensitive if your backend server does not support case-sensitivity. Note: • The Developer Portal does not support case sensitive usernames. • After at least one user has been onboarded into the registry, you cannot change this setting.
User registry managed	Determines whether API Connect manages your user registry. Valid values are: • `true` - select the checkbox • `false` - clear the checkbox
User managed	Determines whether your user registry is writable or not. Select the checkbox to set to `true` for writable. Clear the checkbox for the non-writable option.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Create. The new user registry is displayed in the User registries list.

Adding your custom user registry to the Cloud Manager or API Manager login

To make the custom user registry available for user authentication in the Cloud Manager or the API Manager, you must set it as active in the Settings > User Registries section of the Cloud Manager UI. See Selecting user registries for Cloud Manager and API Manager for more information.
If you want to make the custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Catalog settings > Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see Creating and configuring Catalogs.

Creating a shared custom user registry by using the developer toolkit CLI

You configure a custom user registry by first configuring an integration document that defines your external user registry, and then creating a new custom user registry resource that references the integration document that you just created. You use developer toolkit CLI commands to create the integration document, and then the custom user registry. Finally, to make the registry available for user authentication, you need to configure the registry in the management server, or in the appropriate Catalog, or both, depending on your requirements. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the integration document.

Logging in to the management server CLI

Before you can define the custom user registry, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For full details of the login command, see Logging in to a management server.

For more information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Defining your integration document

You define the configuration of your external user registry in an ***integration_file*.yaml** document, as shown in the following example.

```
integration: '1.0.0'
integration_type: 'user_registry'
info:
  name: 'custom_user_registry'
  version: '1.0.0'
  title: 'Custom External User Registry'
  summary: 'Configure External User Registry'
properties:
  remote: true
secured_endpoint:
  endpoint:  'https://custom.ms.com:8888'
  tls_client_profile_url: {$tslClientProfileUrl}
headers:
  key1: value1
  key2: value2
configuration_schema:
  option_config1:
     type: string
  option_config2:
     type: boolean
```

The integration document properties are described in the following table:

Property	Description
integration_type	The type of integration, which in this case is **user_registry**.
info	Denotes the information section for the integration document.
name	The name of the user registry type, for example ***custom_user_registry***. This name is used in the user registry file to denote the registry type.
version	The version number of the integration document.
title	A descriptive name to display in a graphical user interface.
summary	A brief description of the integration document.
properties	Denotes the property section of the integration document.
remote	Indicates that this is an external user registry. Must be set to **true**.
secured_endpoint	Denotes the external registry endpoint section of the integration document.
endpoint	The secure endpoint of your external user registry interface, where your platform REST API is located. It is made up of the url and port, for example: ***https://custom.ms.com:8888*** API Connect will make REST calls to this endpoint for the user registry actions.
tls_client_profile_url	Optionally set the TLS Client Profile URL that the custom server requires.
headers	Complete the user-defined header section to your external registry specification.
configuration_schema	Complete the user-defined configuration schema section to your external registry specification.

Save your ***integration_file*.yaml** so it can be accessed by the **integrations:create** command in the following section.

Creating your custom integration

To create your custom user registry integration, run the following CLI command:

apic integrations:create --server *mgmt_endpoint_url* --subcollection user-registry *integration_file*.yaml

where:

- ***mgmt_endpoint_url*** is the management platform API endpoint URL.
- ***integration_file*** is the name of the YAML file that defines the configuration of your external user registry.

At this point API Connect makes a REST request call to the **endpoint** that is configured in the integration document, and the external platform REST API will need to return the requested information.
On completion of the integration creation, the command displays the following summary details:

integration_name custom_integration_url

The ***integration_name*** is derived from the **name** property in the configuration YAML file. The ***custom_integration_url*** is the URL with which the integration can be accessed. Both properties are needed when defining your custom user registry in the next section.

After your custom integration has completed successfully, you can continue to create the custom user registry.

Defining your custom user registry

You define the configuration of your custom user registry in a ***custom_config_file*.yaml** file, as shown in the following example.

```
name: 'custom_registry_name'
title: 'display_title'
registry_type: 'custom_user_registry'
integration_url: custom_integration_url
case_sensitive: true_or_false
user_managed: true_or_false
user_registry_managed: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
 - name: provider_name
   title: 'provider_title'
configuration:
  custom_config1: 'value1'
  customize: true
```

The registry properties are described in the following table:

Property	Description
name	The name of the custom user registry. This name is used in the CLI commands.
title	A descriptive name to display in a graphical user interface.
registry_type	The registry type that you configured in the **name** property in the integration document, for example ***custom_user_registry***.
integration_url	The custom integration URL in your API Connect configuration. You can determine the custom integration URL by using the following CLI command: **apic integrations:list --server *mgmt_endpoint_url* --subcollection user-registry** Or you can copy and paste from the summary of the integration creation.
case_sensitive	Determines whether your user registry is case-sensitive. Valid values are: • **true** • **false** To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** set **case_sensitive** to **true** if your backend LDAP server supports case-sensitivity. • Set **case_sensitive** to **false** if your backend LDAP server does not support case-sensitivity. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
user_managed	Determines whether your user registry is writable or not. Must be set to **true** for writable. Set to **false** if you don't want the registry to be writable.
user_registry_managed	Determines whether API Connect manages your user registry. Valid values are: • **true** • **false**
email_required	Determines whether an email address is required as part of the user onboarding process. Valid values are: • **true** • **false** If set to **true**, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
email_unique_if_exist	Determines whether email addresses must be unique within the user registry. Valid values are: • **true** • **false** Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
identity_providers	An array containing the details of your custom server, where: • **name** - is the name of the custom server and is the name that is used in CLI commands • **title** - is the display name of the custom server
configuration	The user-defined configuration based on the **configuration_schema** that is defined in the integration document.

Save your ***custom_config_file*.yaml** so it can be accessed by the **user-registries:create** command in the following section.

Creating your custom user registry

To create your shared custom user registry, run the following CLI command:

apic user-registries:create --server *mgmt_endpoint_url* --org admin *custom_config_file*.yaml

where:

- ***mgmt_endpoint_url*** is the platform API endpoint URL.

- `--org admin` means that the registry will be created in the admin organization, which is required for a user registry to be shared.
- *`custom_config_file`* is the name of the YAML file that defines the configuration of your custom user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The *`registry_name`* is derived from the `name` property in the custom user registry YAML file. The *`registry_url`* is the URL with which the custom registry resource can be accessed.
Your shared custom user registry is now created; see the following sections for instructions on how to make the registry available to users.

Configuring your custom registry for Cloud Manager or API Manager login

If you want to make your custom registry available for authenticating Cloud Manager and API Manager users, you must configure it on the management server.

1. Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   ```
   apic user-registries:list --server mgmt_endpoint_url --org admin
   ```

2. Determine what the current user registries are, because you will need to confirm these as well as add your new custom registry:

   ```
   apic user-registry-settings:get --server mgmt_endpoint_url --output -
   ```

   This command outputs a list of all the current user registries in your environment, similar to the following example:

   ```
   type: user_registry_setting
   api_version: 2.0.0
   name: user-registry-setting
   admin_user_registry_default_url: >-
     https://server.com/api/user-registries/xxxxx/xxxxx-1234
   admin_user_registry_urls:
     - >-
       https://server.com/api/user-registries/xxxxx/xxxxx-1234
   provider_user_registry_default_url: >-
     https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
   provider_user_registry_urls:
     - >-
       https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
   created_at: '2019-09-30T12:22:19.467Z'
   updated_at: '2019-10-17T10:05:37.867Z'
   url: 'https://server.com/api/cloud/settings/user-registries'
   ```

3. Enter the following command to update your user registries (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic user-registry-settings:update --server mgmt_endpoint_url -
   ```

   The following message is output:

   ```
   Reading USER_REGISTRY_SETTING_FILE arg from stdin
   ```

4. If you want to make your custom registry available for authenticating Cloud Manager users, enter the following data, followed by a new line:

   ```
   admin_user_registry_urls:
     - >-
       current_admin_user_registry_urls
     - new_custom_user_registry_url
   ```

   where:
   - *`current_admin_user_registry_urls`* are the current admin user registry URLs, as listed in Step 2 in the `admin_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
   - *`new_custom_user_registry_url`* is the URL of your new custom user registry, as determined in Step 1.
5. If you want to make your custom registry available for authenticating API Manager users, enter the following data, followed by a new line:

   ```
   provider_user_registry_urls:
     - >-
       current_provider_user_registry_urls
     - new_custom_user_registry_url
   ```

   where:
   - *`current_provider_user_registry_urls`* are the current provider organization user registry URLs, as listed in Step 2 in the `provider_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
   - *`new_custom_user_registry_url`* is the URL of your new custom user registry, as determined in Step 1.
6. Press `CTRL D` to terminate the input. A confirmation message is output, for example:

   ```
   user-registry-setting   https://server.com/api/cloud/settings/user-registries
   ```

   where *`server.com`* is your management server endpoint.

Configuring your custom registry in a Catalog

If you want to make your custom registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   ```
   apic user-registries:list --server mgmt_endpoint_url --org admin
   ```

2. Log in to the management server as a member of a provider organization by entering the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

For full details of the `apic login` command, see Logging in to a management server.

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
   ```

   where *`catalog_name`* is the value of the `name` property of the required Catalog. The command returns:

   ```
   Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
   ```

4. Enter the following data, followed by a new line:

   ```
   user_registry_url: custom_registry_url
   ```

   where *`custom_registry_url`* is the URL of your custom registry, obtained in step 1.
5. Press `CTRL D` to terminate the input.

Related information

- Command-line tool reference for the developer toolkit

Configuring an LDAP user registry in the Cloud Manager

You can use the Cloud Manager UI to configure an LDAP user registry as a shared resource to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

Before you begin

To configure an LDAP user registry as a shared resource in the Cloud Manager, the LDAP directory must be created for use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, to authenticate users to the Cloud Manager and the API Manager, or for securing a Catalog to authenticate Developer Portal users.
Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.
One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used across the Cloud Manager, the API Manager, and the Developer Portal components in your environment.

This topic describes how to configure a shared LDAP user registry that is available to all of the provider organizations in your API Connect environment. If you want to create an organization-specific registry, see Creating an LDAP user registry in API Manager for more information. Note also that the visibility of a user registry is set to **`shared`** by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see Setting visibility for a user registry.
Note:

- If you configure your LDAP user registry to be writable (by selecting the User Managed check box on the registry), you can use the Developer Portal UI for onboarding and authenticating new Developer Portal users, as well as those users that already exist in the LDAP database. A writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
- You can also create and manage LDAP user registries by using the developer toolkit CLI (see Using the CLI to configure a shared LDAP user registry), and by using the API Connect REST APIs (see the API Connect REST API documentation).
- You can map external LDAP groups to API Connect user roles to enable greater control of user access, but this configuration can be done only by using the developer toolkit CLI; see Using the CLI to configure a shared LDAP user registry for details.

You create an LDAP user registry by configuring a set of properties in the Cloud Manager UI. If you want to enable writable LDAP, you must complete the Attribute Mapping section by selecting the User Managed checkbox, and providing the mapping of your source LDAP attribute names to the target API Connect values. You can also change a registry to be read-only again by clearing the User Managed checkbox. After configuring the user registry, you must set it as active in Settings > User Registries. To make the registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions.

For general information about authenticating with LDAP, see LDAP authentication.

Procedure

Follow these steps to configure a new LDAP user registry as a shared resource in the Cloud Manager UI.

1. In the Cloud Manager, click Resources.
2. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the

catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

3. Select LDAP User Registry for the user registry type, and enter the following information:

Field	Description
Title	Enter a descriptive name to display on the screen.
Name	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the **`Title`** of the User Registries when rendering them at the login page, rather than the **`Display Name`**.
Summary (optional)	Enter a brief description.
Address	Enter the IP address or host name of the LDAP server.
Port	Enter the Port number that API Connect can use to communicate with the LDAP registry. For example, `389`.
Select a TLS Client Profile (optional)	Select the TLS Client Profile that the LDAP server requires.
Select an LDAP protocol version	Select the version number for the LDAP protocol that you are using.
Remote directory is Microsoft Active Directory	Select this option if you use Active Directory.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** select Case sensitive if your backend LDAP server supports case-sensitivity. • Do **not** select Case sensitive if your backend LDAP server does not support case-sensitivity. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Enable External Group Mapping	Enable this property if you want user to use this user registry to map external LDAP groups to API Connect user roles to enable more control of user authorization.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Next and enter the authentication information, which will vary depending on the selected Authentication Method. The choices are:
   - Compose DN - Select this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, **`uid=<username>,ou=People,dc=company,dc=com`** is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, Compose DN is not supported with the DataPower API Gateway.
   - Compose UPN - Select this format if your LDAP directory supports binding with User Principal Names such as **`john@acme.com`**. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.
     Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
   - Search DN - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.
     Optionally select a scope to specify which part of the directory information tree is examined:
     - Whole subtree (default)
     - Base object
     - Single level

   For all of the authentication methods:

   If you are creating an LDAP registry to authenticate users of an API, you can specify an LDAP authorization group to restrict API access. To be able to call an API that is secured by the LDAP registry, a user must successfully authenticate with their LDAP user ID and password **and** they must be a member of the specified authorization group. The authorization group can be a Static Group or Dynamic Group. A static group is one in which the individual members of the group are explicitly listed. A dynamic group is one which is defined according to the set of attributes that the group members share in common.

5. For authentication method Compose DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example `(uid=`.
Suffix	Specify the suffix to the DN. For example `)`.
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

6. For authentication method Compose UPN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.

Field	Description
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Suffix	Enter the domain part of the user principal name. For example, `@acme.com`.
Use group authentication (optional)	Enter the Filter condition for the group.

7. For authentication method Search DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example `(uid=`.
Suffix	Specify the suffix to the DN. For example `)`.
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

8. Optional: Click Test configuration to test the settings for your LDAP user registry. Enter valid credentials to ensure that you can access the LDAP database.
9. Optional: If you want to make your LDAP user registry writable, select the User Managed checkbox in the Attribute Mapping section, and provide the mapping of your source LDAP attribute names to the target API Connect values. Click Add to add each name/value pair, specified as follows:
   - LDAP ATTRIBUTE NAME - is the name of the source LDAP attribute.
   - API CONNECT VALUE - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in `[ ]` with the value that the user supplies when signing up.

   The default user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`, as shown in the following example:

LDAP ATTRIBUTE NAME	API CONNECT VALUE
`dn`	`uid=[username],ou=users,dc=company,dc=com`
`cn`	`[first_name] [last_name]`
`sn`	`[last_name]`
`mail`	`[email]`
`userPassword`	`[password]`

   You must ensure that you enter the correct attribute mapping values for your LDAP configuration, to enable API Connect to access the LDAP database. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
10. Click Create.
    Your new LDAP registry is shown in the list of User Registries on the Resources page.

What to do next

To make the LDAP registry available for user authentication in the Cloud Manager and the API Manager, you must set it as active in the Settings > User Registries section. See Selecting user registries for Cloud Manager and API Manager for more information.
If you want to make the LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Settings > Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see Creating and configuring Catalogs.

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see Creating a basic authentication security definition.
- To use for authentication in the User Security configuration for a native OAuth provider, see Configuring user security for a native OAuth provider.

- **Using the CLI to configure a shared LDAP user registry**
  You can use the developer toolkit CLI to configure an LDAP user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

Related concepts

- Managing authentication and security
- User registries overview

Related tasks

- Configuring a Local User Registry
- Configuring an Authentication URL user registry
- Setting visibility for a user registry

Related information

- LDAP authentication
- Securing your API Connect Cloud with LDAP (series of developer articles)

Using the CLI to configure a shared LDAP user registry

You can use the developer toolkit CLI to configure an LDAP user registry to provide user authentication for the Cloud Manager, the API Manager, and the Developer Portal. APIs can also be secured with an LDAP user registry.

Before you begin

To configure an LDAP user registry as a resource in the Cloud Manager, the LDAP directory must be created and available to use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for authenticating users to the Cloud Manager, the API Manager, and the Developer Portal.

Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.
One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for onboarding and authenticating Developer Portal users, while a shared LDAP user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure a shared LDAP user registry. If you want to create an organization-specific registry, see Using the CLI to create an organization-specific LDAP user registry for more information. Note also that the visibility of a user registry is set to **`shared`** by default. However, you can change the visibility setting to make the registry private, or visible only to specific provider organizations. For more information, see Setting visibility for a user registry.
Note:

- If the LDAP user registry is configured as writable (by enabling the **`user-managed`** property on the registry), you can use the Developer Portal UI for onboarding and authenticating new Developer Portal users, as well as those users that already exist in the LDAP database. A writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
- You can also create LDAP user registries by using the Cloud Manager UI (see Configuring an LDAP user registry in the Cloud Manager), and by using the API Connect REST APIs (see the API Connect REST API documentation).
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable more control of user authorization by using the developer toolkit CLI. The following instructions explain how to set the **`external_group_mapping_enabled`** configuration on your LDAP user registry resource. For information about how to set external role mapping on your API Connect user roles, see Configuring LDAP group mappings on Cloud Manager user roles.

You configure an LDAP user registry by first defining the registry details in a configuration file. You then use a developer toolkit CLI command to create the registry, passing the configuration file as a parameter. Finally, to make the registry available for user authentication, you need to configure the registry in the management server, or in the appropriate Catalog, or both, depending on your requirements. To secure APIs with an LDAP registry, you must configure security definitions. You can use the following instructions to create a writable or a read-only LDAP user registry.

For general information about authenticating with LDAP, see LDAP authentication.

Logging in to the management server CLI

Before you can define the LDAP user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the **`--realm`** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **`title`** value should enable you to determine which identity provider to use; you can then copy the corresponding **`--realm`** parameter directly from the displayed **`realm`** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **`default-idp-1`**.
For full details of the login command, see Logging in to a management server.

For more information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Defining your LDAP configuration

You define the configuration of your LDAP user registry in an ***`ldap_config_file`*`.yaml`** file, as shown in the following example. Note that the actual contents of your YAML file will vary depending on the authentication method of your LDAP server, and this is explained in the following tables.

```
name: registry_name
title: "display_title"
integration_url: LDAP_integration_url
```

```
user_managed: true_or_false
user_registry_managed: false
directory_type: ad
external_group_mapping_enabled: true_or_false
case_sensitive: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
 - name: provider_name
   title: provider_title
endpoint:
  endpoint: "ldap_server_url_and_port"
configuration:
  authentication_method: authentication_method
  authenticated_bind: "true_or_false"
  admin_dn: "admin_dn"
  admin_password: admin_password
  search_dn_base: "search_dn_base"
  search_dn_scope: search_dn_scope
  search_dn_filter_prefix: prefix
  search_dn_filter_suffix: suffix
  attribute_mapping:
    dn: "distinguished_name"
    cn: "common_name"
    sn: "last_name"
    mail: "email_address"
    userPassword: "password"
```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
`name`	The name of the registry. This name is used in CLI commands.
`title`	A descriptive name to display in a graphical user interface.
`integration_url`	The LDAP integration URL in your API Connect configuration. You can determine the LDAP integration URL by using the following CLI command: `apic integrations:list --server mgmt_endpoint_url --subcollection user-registry`
`user_managed`	Determines whether your user registry is writable or not. Must be set to `true` for writable LDAP. You can change this setting to `false` if you don't want the registry to be writable; see the Switching your LDAP registry between writable and read-only section at the end of this topic for details. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
`user_registry_managed`	Must be set to `false` for LDAP. Determines whether API Connect manages your user registry. Only LUR registries are managed by API Connect.
`directory`	Specify one of the of following values for the directory type: • `ad` - Indicates that the LDAP is Microsoft Active Directory • `standard` - (Default value) Indicates any other LDAP
`external_group_mapping_enabled`	Determines whether your user registry supports LDAP group mapping. Valid values are: • `true` • `false` The default value is `false`.
`case_sensitive`	Determines whether your user registry is case-sensitive. Valid values are: • `true` • `false` To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** set `case_sensitive` to `true` if your backend LDAP server supports case-sensitivity. • Set `case_sensitive` to `false` if your backend LDAP server does not support case-sensitivity. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
`email_required`	Determines whether an email address is required as part of the user onboarding process. Valid values are: • `true` • `false` If set to `true`, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
`email_unique_if_exist`	Determines whether email addresses must be unique within the user registry. Valid values are: • `true` • `false` Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
`identity_providers`	An array containing the details of your LDAP server, where: • `name` - is the name of the LDAP server and is the name that is used in CLI commands • `title` - is the display name of the LDAP server
`endpoint`	The endpoint of your LDAP server, made up of the url and port, for example: `"ldap://server.com:389"`
`tls_profile`	Optionally set the TLS Client Profile that the LDAP server requires.

Property	Description
`protocol_version`	Optionally set the version number for the LDAP protocol that you are using. Valid values are: • `2` • `3` Defaults to `3` if not explicitly set.

The properties in the configuration section will vary depending on the selected authentication method. The three authentication methods are:

- `compose_dn` - Set this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, `compose_dn` is not supported with the DataPower API Gateway.
- `compose_upn` - Set this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.
  Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
- `search_dn` - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For authentication method `compose_dn`, set the following configuration properties:

Properties	Description
`authentication_method`	`compose_dn`
`authenticated_bind`	The bind method. Valid values are: • `"true"` - authenticated bind • `"false"` - anonymous bind If specific permissions are not needed to search the registry, select `"false"`. If specific permissions are necessary, select `"true"`.
`admin_dn`	If `authenticated_bind` is set to `"true"`, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: `"cn=admin,dc=company,dc=com"`
`admin_password`	If `authenticated_bind` is set to `"true"`, enter the user password for the `admin_dn`.
`search_dn_base`	Optionally set a base DN, for example: `"dc=company,dc=com"`
`bind_prefix`	Set the prefix to the DN, for example: `(uid=`
`bind_suffix`	Set the suffix to the DN, for example: `)`

`attribute_mapping`	If `user_managed` is set to `true`, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:
	`ldap_registry_attribute_name``: "`*`apic_ldap_attribute_value`*`"`
	Where:
	• *ldap_registry_attribute_name* - is the name of the source LDAP attribute
	• *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in `[ ]` with the value that the user supplies when signing up.
	The user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`. The following extract shows an example of an attribute mapping:
	`attribute_mapping:`
	`  dn: "uid=[username],ou=users,dc=company,dc=com"`
	`  cn: "[first_name] [last_name]"`
	`  sn: "[last_name]"`
	`  mail: "[email]"`
	`  userPassword: "[password]"`

For authentication method `compose_upn`, set the following configuration properties:

Properties	Description
`authentication_method`	`compose_upn`
`authenticated_bind`	The bind method. Valid values are: • `"true"` - authenticated bind • `"false"` - anonymous bind If specific permissions are not needed to search the registry, select `"false"`. If specific permissions are necessary, select `"true"`.
`admin_dn`	If `authenticated_bind` is set to `"true"`, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: `"cn=admin,dc=company,dc=com"`
`admin_password`	If `authenticated_bind` is set to `"true"`, enter the user password for the `admin_dn`.
`bind_suffix`	Enter the domain part of the user principal name. For example: `@acme.com`

Properties	Description

`attribute_mapping`	If `user_managed` is set to `true`, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:
	
	`ldap_registry_attribute_name: "apic_ldap_attribute_value"`
	
	Where:
	
	• *ldap_registry_attribute_name* - is the name of the source LDAP attribute
	• *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in `[ ]` with the value that the user supplies when signing up.
	
	The user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`. The following extract shows an example of an attribute mapping:
	
	`attribute_mapping:`
	`  dn: "uid=[username],ou=users,dc=company,dc=com"`
	`  cn: "[first_name] [last_name]"`
	`  sn: "[last_name]"`
	`  mail: "[email]"`
	`  userPassword: "[password]"`

For authentication method `search_dn`, set the following configuration properties:

Property	Description
`authentication_method`	`search_dn`
`authenticated_bind`	The bind method. Valid values are: • `"true"` - authenticated bind • `"false"` - anonymous bind If specific permissions are not needed to search the registry, select `"false"`. If specific permissions are necessary, select `"true"`.
`admin_dn`	If `authenticated_bind` is set to `"true"`, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: `"cn=admin,dc=company,dc=com"`
`admin_password`	If `authenticated_bind` is set to `"true"`, enter the user password for the `admin_dn`.
`search_dn_base`	Optionally set a base DN, for example: `"dc=company,dc=com"`
`search_dn_scope`	Optionally set the search DN scope. The scope determines which part of the directory information tree is examined. Possible values are: • `base` • `one` • `sub` (default value)
`search_dn_filter_prefix`	Set the prefix to the DN, for example: `(uid=`
`search_dn_filter_suffix`	Set the suffix to the DN, for example: `)`

`attribute_mapping`	If `user_managed` is set to `true`, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows:
	
	`ldap_registry_attribute_name: "apic_ldap_attribute_value"`
	
	Where:
	
	• *ldap_registry_attribute_name* - is the name of the source LDAP attribute
	• *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in `[ ]` with the value that the user supplies when signing up.
	
	The user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`. The following extract shows an example of an attribute mapping:
	
	`attribute_mapping:`
	`  dn: "uid=[username],ou=users,dc=company,dc=com"`
	`  cn: "[first_name] [last_name]"`
	`  sn: "[last_name]"`
	`  mail: "[email]"`
	`  userPassword: "[password]"`

Save your `ldap_config_file.yaml` so it can be accessed by the `user-registries:create` command in the following section. See the Example section for an example configuration file.

Creating your LDAP user registry

To create your shared LDAP user registry, run the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org admin ldap_config_file.yaml
```

where:

- `mgmt_endpoint_url` is the platform API endpoint URL.
- `--org admin` means that the registry will be created in the admin organization, which is required for a user registry to be shared.
- `ldap_config_file` is the name of the YAML file that defines the configuration of your LDAP user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

The *registry_name* is derived from the `name` property in the configuration YAML file. The *registry_url* is the URL with which the registry resource can be accessed. Your shared LDAP user registry is now created; see the following sections for instructions on how to make the registry available to users.

Configuring your LDAP registry for Cloud Manager or API Manager login

If you want to make your LDAP registry available for authenticating Cloud Manager and API Manager users, you must configure it on the management server.

1. Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   ```
   apic user-registries:list --server mgmt_endpoint_url --org admin
   ```

2. Determine what the current user registries are, because you will need to confirm these as well as add your new LDAP registry:

   ```
   apic user-registry-settings:get --server mgmt_endpoint_url --output -
   ```

   This command outputs a list of all the current user registries in your environment, similar to the following example:

   ```
   type: user_registry_setting
   api_version: 2.0.0
   name: user-registry-setting
   admin_user_registry_default_url: >-
     https://server.com/api/user-registries/xxxxx/xxxxx-1234
   admin_user_registry_urls:
     - >-
       https://server.com/api/user-registries/xxxxx/xxxxx-1234
   provider_user_registry_default_url: >-
     https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
   provider_user_registry_urls:
     - >-
       https://https://server.com/api/user-registries/xxxxx/xxxxx-5678
   created_at: '2019-09-30T12:22:19.467Z'
   updated_at: '2019-10-17T10:05:37.867Z'
   url: 'https://server.com/api/cloud/settings/user-registries'
   ```

3. Enter the following command to update your user registries (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic user-registry-settings:update --server mgmt_endpoint_url -
   ```

   The following message is output:

   ```
   Reading USER_REGISTRY_SETTING_FILE arg from stdin
   ```

4. If you want to make your LDAP registry available for authenticating Cloud Manager users, enter the following data, followed by a new line:

   ```
   admin_user_registry_urls:
     - >-
       current_admin_user_registry_urls
     - new_ldap_user_registry_url
   ```

   where:
   - *current_admin_user_registry_urls* are the current admin user registry URLs, as listed in Step 2 in the `admin_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
   - *new_ldap_user_registry_url* is the URL of your new LDAP user registry, as determined in Step 1.

5. If you want to make your LDAP registry available for authenticating API Manager users, enter the following data, followed by a new line:

   ```
   provider_user_registry_urls:
     - >-
       current_provider_user_registry_urls
     - new_ldap_user_registry_url
   ```

   where:
   - *current_provider_user_registry_urls* are the current provider organization user registry URLs, as listed in Step 2 in the `provider_user_registry_urls` section. Note that you must include all of the user registry URLs that you want to remain, listing each URL on a new line.
   - *new_ldap_user_registry_url* is the URL of your new LDAP user registry, as determined in Step 1.

6. Press `CTRL D` to terminate the input. A confirmation message is output, for example:

   ```
   user-registry-setting   https://server.com/api/cloud/settings/user-registries
   ```

   where *server.com* is your management server endpoint.

Configuring your LDAP registry in a Catalog

If you want to make your LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   ```
   apic user-registries:list --server mgmt_endpoint_url --org admin
   ```

2. Log in to the management server as a member of a provider organization by entering the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   For full details of the `apic login` command, see Logging in to a management server.

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
```

where *catalog_name* is the value of the `name` property of the required Catalog. The command returns:

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

4. Enter the following data, followed by a new line:

```
user_registry_url: ldap_registry_url
```

where *ldap_registry_url* is the URL of your LDAP registry, obtained in step 1.
5. Press `CTRL D` to terminate the input.

Switching your LDAP registry between writable and read-only

After an LDAP user registry has been created, it can be switched between writable and read-only by updating the `user_managed` property in the registry configuration. Complete the following steps.

1. Determine the name or ID of the LDAP user registry that you want to update, by running the following command (or you can use the summary from the registry creation):

```
apic user-registries:list --server mgmt_endpoint_url --org admin
```

The command returns a list of all the user registries on that server, shown by name followed by their registry URL. The registry ID is located at the end of the URL, for example `https://company.com/api/user-registries/x-x-x-x-x/registry_id`.
2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

```
apic user-registries:update --server mgmt_endpoint_url --org admin registry_name_or_id -
```

where `registry_name_or_id` is the name or ID of the LDAP user registry that you want to update (as determined in the previous step). The command returns:

```
Reading USER_REGISTRY_FILE arg from stdin
```

3. Enter the following data, followed by a new line:

```
user_managed: true_or_false
```

where `true` makes the registry writable, and `false` makes the registry read-only.
4. Press `CTRL D` to terminate the input.

Note that if you are changing your registry from read-only to writable, you must also set the `attribute_mapping` configuration, as described in the previous registry property tables.

Using an LDAP user registry to secure APIs

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see Creating a basic authentication security definition.
- To use for authentication in the User Security configuration for a native OAuth provider, see Configuring user security for a native OAuth provider.

For details of all the `apic user-registries` and `apic configured-catalog-user-registries` commands, see apic user-registries and apic configured-catalog-user-registries.

Example

The following example shows a configuration file that uses the Search DN authentication method for setting up writable LDAP:

```
name: sdn-ldap
title: "SDN LDAP User Registry"
integration_url: https://mycompany.com/api/cloud/integrations/user-registry/xxx-xxx-xxx
user_managed: true
user_registry_managed: false
directory_type: standard
case_sensitive: false
identity_providers:
  - name: ldap
    title: "SDN LDAP Identity Provider"
endpoint:
  endpoint: "ldap://mycompany.com:389"
configuration:
  authentication_method: search_dn
  authenticated_bind: "true"
  admin_dn: "cn=admin,dc=company,dc=com"
  admin_password: xxxx
  search_dn_base: "dc=company,dc=com"
  search_dn_scope: sub
  search_dn_filter_prefix: (uid=
  search_dn_filter_suffix: )
  attribute_mapping:
    dn: "uid=[username],ou=users,dc=company,dc=com"
    cn: "[first_name] [last_name]"
    sn: "[last_name]"
    mail: "[email]"
    userPassword: "[password]"
```

Related information

- Command-line tool reference for the developer toolkit
- Securing your API Connect Cloud with LDAP (series of developer articles)

Configuring a Local User Registry

A Local User Registry (LUR) can be configured to provide user authentication for both Cloud Manager and API Manager.

About this task

Local User Registries (LURs) are the default user registries included in API Connect. LURs are local databases included with API Connect. Two default LURs are installed and configured during installation of API Connect. They cannot be deleted. The default Admin user account is stored in the Provider LUR.

You can create and configure a new LUR. After you create it, you must set it as active in Settings > User Registries. See Selecting user registries for Cloud Manager and API Manager.

One of the following roles is required to configure user registries:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

Procedure

Follow these steps to configure a new LUR:

1. In the Cloud Manager, click .
2. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
3. Select Local User Registry as the type for the user registry and enter the following information:

Field	Description
Title (required)	Enter a descriptive name for use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the **`Title`** of the User Registries when rendering them at the login page, rather than the **`Display Name`**.
Summary (optional)	Enter a brief description.
Case sensitive	Select this setting if user names are case-sensitive. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. For new Local User Registries, this setting is always selected; so if email addresses are contained in the user record, they must be unique. However, for existing Local User Registries this setting can be edited. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Save.

Results

The user registry is saved and is available to be configured for user authentication in Cloud Manager or API Manager by selecting it in Settings > User Registries. See Selecting user registries for Cloud Manager and API Manager.

Related concepts

- Managing authentication and security
- User registries overview

Related tasks

- Configuring an Authentication URL user registry
- Configuring an LDAP user registry in the Cloud Manager

- Setting visibility for a user registry

Configuring an OIDC user registry in Cloud Manager

Configure a shared OIDC user registry for user onboarding and authentication when multi-factor authentication (MFA) is required.

You can create an OIDC user registry that is specific to a provider organization, or that is shared and available to all the provider organizations in your API Connect environment. An organization-specific OIDC user registry is used for onboarding and authenticating Developer Portal users, while a shared OIDC user registry can be used for onboarding and authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to create a shared registry. For information on how to create an organization-specific registry, see Creating an OIDC user registry in API Manager.

Important: If you are using X (formerly known as Twitter) as your OIDC provider, API Connect supports only the OAuth 1.0a method, so your X application must be configured to use OAuth 1.0a. Other OIDC providers support OAuth 2.0.
API Connect provides two methods for creating an OIDC user registry in Cloud Manager, as described in the following sections:

- Using the UI to configure an OIDC user registry
- Using the CLI to configure an OIDC user registry

Using the UI to configure an OIDC user registry

Use the Cloud Manager application's user interface to configure a shared OIDC user registry when multi-factor authentication (MFA) is required.

1. In the Cloud Manager navigation pane, click Resources.
2. Click User Registries.
3. Click Create and select OIDC User Registry.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
4. On the Create OIDC User Registry page, use the fields in each of the following sections to configure the registry settings, and then click Create.
   Many of the registry settings are preconfigured to simplify the configuration steps.

   Provider Type
   : Use the settings in Table 1 to define the provider type.

Table 1. Provider Type settings

Field	Description
Provider Type	OIDC provider. Select one of the following supported OIDC providers: • Facebook • GitHub • Google • LinkedIn • Slack • Twitter • Windows Live • Standard OIDC (default value allows you to specify another provider)
Title	Provide a descriptive name for display purposes.
Name	Automatically generated. This name is used in CLI commands to reference the registry. For details of the CLI commands for managing user registries, see the apic user-registries topic in the Command Line tool reference section of this documentation.
Summary	Provide a brief description of the new registry.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

   Provider Endpoint
   : Automatically generated for most supported providers. In the Authorization Endpoint field, type the URL of the provider's authorization endpoint.

   Token Endpoint
   : Fill in the settings as described in Table 2.

Table 2. Token Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's token endpoint.
TLS	Select the TLS Client Profile for the token endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

   UserInfo Endpoint
   : Fill in the settings as described in Table 3.

Table 3. UserInfo Endpoint settings

Field	Description

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's userinfo endpoint. For LinkedIn, the endpoint changed and now uses the following value: `https://api.linkedin.com/v2/me` If you previously configure the OIDC registry using the legacy LinkedIn OIDC protocol, you do not need to specify this endpoint.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

Email Endpoint

Required for the legacy LinkedIn OIDC protocol.
Due to changes in LinkedIn, the configuration settings for the OIDC registry in API Connect also changed. If you are configuring a new OIDC registry for LinkedIn, you can leave the Email Endpoint field blank.

If you previously configured an OIDC registry for use with the legacy LinkedIn OIDC protocol, update your configuration by providing the Email endpoint settings as described in Table 4. The new setting is required for the legacy LinkedIn OIDC protocol.

Table 4. Email Endpoint settings

Field	Description
URL	For the legacy LinkedIn OIDC protocol, this field defaults to the following value: `https://api.linkedin.com/v2/emailAddress?q=members&projection=(elements*(handle~))` You can override the setting by typing in a new value.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

JWKS Endpoint

Fill in the settings as described in Table 5.

Table 5. JWKS Endpoint settings

Field	Description
URL	Type the URL of the read-only endpoint that contains the public keys' information in JWKS format.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

Logout

Fill in the settings as described in Table 6.

Table 6. Logout

Field	Description
Logout redirect	Optionally, give a logout redirect URL, allowing APIC to initiate an https 302 redirect to the configured endpoint. Note: When the Developer Portal uses this feature, it does not retain control afterwards. Instead, the content is presented from the redirected endpoint.

Client Information

Fill in the settings as described in Table 7.

Table 7. Client Information settings

Field	Description
Client ID	Provide the client ID of the application that is registered with the selected OIDC provider.
Client Secret	Provide the client secret of the application that is registered with the selected OIDC provider. This field is optional when you create an OIDC directory of type Standard and set the Client Authentication Method to Data encoded form body. If you do not use the Client Secret, make sure that mutual TLS is possible with the OIDC provider.
Response Type	Preconfigured for most of the supported OIDC providers. Specify the data type of the response that will be received from the OIDC provider.
Scopes	Preconfigured for most of the supported OIDC providers. Specify the access scope for the OIDC provider.
Client Authentication Method	Preconfigured for most of the supported OIDC providers. Select the authentication method to be used with the OIDC provider. Options are: • HTTP Basic authentication schema • Data encoded form body

Additional Support

Optional. Select the additional security parameters described in Table 8.

Table 8. Additional security options

Security parameter	Setting name for CLI	Description
NONCE	`nonce`	Enable the NONCE extension to prevent compromised requests from being used again (replayed).
Proof Key for Code Exchange (PKCE)	`pkce`	Enable the PKCE extension to allow public clients to mitigate the threat of having the authorization code intercepted.

Advanced Features

Optional. Select the advanced features described in Table 9.

Table 9. Advanced features

Feature / UI Label	Setting name for CLI	Description
Auto onboard	`auto_onboard`	Allow users to executes calls to APIs without logging in first, provided they present a valid token issued by the OIDC provider. Note: Does not apply to consumer organizations.
Always use the userinfo endpoint	`userinfo`	Configures the OIDC user registry to always fetch user data from the userinfo endpoint, if populated.
Use Portal as Endpoint for external OIDC provider traffic	`proxy_redirect`	When authenticating users, redirect the external OIDC provider to communicate with the Developer Portal instead of API Manager.

Feature / UI Label	Setting name for CLI	Description
Return third-party access token	`proxy_access_token`	Include the third-party OIDC access token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an ERR_HTTP2_PROTOCOL_ERROR or ERR_CONNECTION_CLOSED error.
Return third-party id_token	`proxy_id_token`	Include the third-party OIDC id_token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an ERR_HTTP2_PROTOCOL_ERROR or ERR_CONNECTION_CLOSED error.
Token Relay	`token_relay`	Allow `access_token/refresh_token` to send in 302 redirect for logout
Userinfo POST	`post_userinfo`	If supported by your OIDC provider, by using HTTP POST method when contacting userinfo endpoint. Note: Not all OIDC providers support POST. Ensure that your OIDC provider supports this before you enable the feature.
Use IBM APIC token expiration setting from cloud	`override_provider_ttl`	Override the OIDC provider's token expiration setting with the access token and refresh token expiration settings that are configured in API Connect. For information on configuring the access token and refresh token expiration settings in API Connect, see Configuring timeouts for access tokens and refresh tokens.
Disable hash verification (CLI only)	`disable_hash_verification`	Disable `at_hash`, `c_hash`

User Mapping

Fill in the settings as described in Table 10.
Note: The User Mapping fields are preconfigured for most of the supported OIDC providers to minimize potential errors; use care when changing the settings. For the Standard OIDC option, contact your OIDC provider to obtain the details of the fields.

Table 10. User mapping settings

Field	Description
Username	The name of the field in the response token that contains the user's user name. Note: The Username field must be unique for this OIDC registry, because it identifies the user in the system and cannot be changed.
Email	The name of the field in the response token that contains the user's email address.
First name	The name of the field in the response token that contains the user's given name.
Last name	The name of the field in the response token that contains the user's surname.

Redirect URI

The Redirect URI section lists the endpoints to which the OIDC authorization server will redirect the authorization code. There is one endpoint for each API Connect organization type: Cloud administration, Provider organization, and Consumer organization. The redirect endpoint is required when a user registers their application with the OIDC provider. For example, if this OIDC user registry is used by a provider organization, the user must register the provider organization's redirect endpoint with the OIDC provider. These read-only fields are provided for you to copy the endpoint values as required.

Enabling the OIDC user registry

Complete the following steps to enable the new user registry for Cloud Manager, API Designer, or both.

1. On the navigation pane, click Settings.
2. Click User Registries > Edit.
3. Click Edit in the section that corresponds to the application for which you are enabling the new user registry.
4. Select the new OIDC user registry.
5. When you have finished enabling the registry, click Save.

Using the CLI to configure an OIDC user registry

Use the developer toolkit CLI to configure a shared OIDC user registry when multi-factor authentication (MFA) is required.

Configure an OIDC user registry by first defining the registry details in a configuration file. You then use a CLI command to create the registry, passing the configuration file as parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated catalog.
Note:

- An OIDC registry cannot be used to secure APIs on the gateway.
- Because the interaction with the third-party OIDC provider is handled by the Management server, the Management server is the application from the point of view of the third-party OIDC provider. Your OIDC redirection endpoint is used by the authorization server to send the token to the Management server, so it must be accessible to the OIDC provider through your firewall. When you register your application with the third-party OIDC provider, you are required to supply the associated OIDC redirect URI, `https://consumer.mycompany.com/consumer-api/oauth2/redirect` for example. However, this information is not available until you have created your OIDC user registry in API Connect. You must first register your application without this information and then update it later, as detailed in the instructions on this page.

Logging in to the Management server

Before you can create an OIDC user registry, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For full details of the **apic login** command, see Logging in to a management server.

Defining your OIDC registry configuration

Define the configuration of your OIDC user registry in a YAML file. At a minimum, the YAML file must include the settings from the following list. For additional settings, see the tables provided with the UI version of the procedure in this topic.

```
title: registry_title
integration_url: oidc_integration_url
case_sensitive: case_sensitivity_setting
email_required: true_or_false
email_unique_if_exist: true_or_false
configuration:
  client_id: 'app_client_id'
  client_secret: 'my-client-secret'
  provider_type: oidc_provider_type
```

where:

- *registry_title* is your chosen descriptive title for the user registry.
- *oidc_integration_url* is the OIDC integration URL in your API Connect configuration. You can determine the OIDC integration URL by using the following CLI command:

  ```
  apic integrations:list --server mgmt_endpoint_url --subcollection user-registry
  ```

- *case_sensitivity_setting* determines whether your user registry is case-sensitive. Valid values are:
  - **true**
  - **false**

  To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on the backend OIDC provider:
  - Only set **case_sensitive** to **true** if the backend OIDC provider supports case-sensitivity.
  - Set **case_sensitive** to **false** if the backend OIDC provider does not support case-sensitivity.

  Note: After at least one user has been added to the registry, you cannot change this setting.
- *email_required* determines whether an email address is required as part of the user onboarding process. Valid values are:
  - **true**
  - **false**

  If set to **true**, the source identity provider must supply the email address as part of the authentication process during onboarding.
  Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
- *email_unique_if_exist* determines whether email addresses must be unique within the user registry. Valid values are:
  - **true**
  - **false**

  Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
- *app_client_id* is the client ID of the application that is registered with the OIDC server, and must be in string format.
- *my-client-secret* is the client secret of the application that is registered with the OIDC server, and must be in string format.
- *oidc_provider_type* is the type of OIDC provider; specify one of the following values:
  - **facebook**
  - **github**
  - **google**
  - **linkedin**
  - **slack**
  - **twitter**
  - **windows_live**
  - **standard**

    Note:
    - Use the **twitter** provider type if your OIDC provider is X (formerly known as Twitter).
    - Use the **standard** provider type for any OIDC provider that is compliant with the OIDC standard.
      If the provider type is **standard**, you must include the following additional properties in the **configuration** section of your YAML file:

      ```
      authorization_endpoint: 'oidc_auth_endpoint'
      token_endpoint:
        endpoint: 'oidc_token_endpoint'
      ```

      where:
      - *oidc_auth_endpoint* is the authorization endpoint on the OIDC server, and must be in string format.
      - *oidc_token_endpoint* is the token endpoint on the OIDC server, and must be in string format.

Default OIDC configurations

For each OIDC provider type, API Connect assumes a default configuration, but you can override the default configuration properties in your YAML file.

Note: Endpoints are subject to change; when you create a registry you must verify the default endpoint provided by API Connect.
API Connect makes the best effort to update endpoints for the known OIDC user registry types. However the OIDC provider can change the endpoint at any time. It is the customer's responsibility to confirm that the endpoints provided by API Connect are supported by the OIDC provider and to update the OIDC configuration as needed.

The default configurations are as follows:

- Facebook

```
authorization_endpoint: 'https://www.facebook.com/v3.1/dialog/oauth'
token_endpoint:
  endpoint: 'https://graph.facebook.com/v3.1/oauth/access_token'
userinfo_endpoint:
  endpoint: 'https://graph.facebook.com/me'
scope: email public_profile
field_mapping:
  username: email
  email: email
  last_name: last_name
  first_name: first_name
```

- GitHub

```
authorization_endpoint: 'https://github.com/login/oauth/authorize'
token_endpoint:
  endpoint: 'https://github.com/login/oauth/access_token'
userinfo_endpoint:
  endpoint: 'https://api.github.com/user'
scope: 'read:user user:email'
field_mapping:
  username: login
  email: email
  last_name: name
  first_name: name
```

- Google

```
authorization_endpoint: 'https://accounts.google.com/o/oauth2/v2/auth'
token_endpoint:
  endpoint: 'https://www.googleapis.com/oauth2/v4/token'
scope: openid profile email
field_mapping:
  username: email
  email: email
  last_name: family_name
  first_name: given_name
```

- LinkedIn
  Some settings differ between the legacy configuration and the newer configuration. Be sure to use the correct settings for the LinkedIn protocol that you are using.

  - New OIDC protocol:

```
authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
  credential_location: form_body
  field_mapping:
    email: email
    username: name
    first_name: given_name
    last_name: family_name
  scope: openid profile email
  token_endpoint:
    endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
  userinfo_endpoint:
    endpoint: 'https://api.linkedin.com/v2/me'
```

  - Legacy OIDC protocol:

```
authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
token_endpoint:
  endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
userinfo_endpoint:
  endpoint: 'https://api.linkedin.com/v1/people/~:(id,first-name,last-name,picture-url,public-profile-url,email-
address)?format=json'
scope: r_basicprofile r_emailaddress
field_mapping:
  username: emailAddress
  email: emailAddress
  last_name: lastName
  first_name: firstName
credential_location: form_body
```

- Slack

```
authorization_endpoint: 'https://slack.com/oauth/authorize'
token_endpoint:
  endpoint: 'https://slack.com/api/oauth.access'
userinfo_endpoint:
  endpoint: 'https://slack.com/api/users.identity'
scope: identity.basic identity.email
field_mapping:
  username: user.email
  email: user.email
  last_name: user.name
  first_name: user.name
```

- X (formerly known as Twitter)
  Although Twitter re-branded to X, the URL for the OIDC endpoint remains twitter.com.

```
request_endpoint: https://api.twitter.com/oauth/request_token'
authorization_endpoint: https://api.twitter.com/oauth/authenticate'
token_endpoint:
    endpoint: 'https://api.twitter.com/oauth/access_token'
```

```
userinfo_endpoint:
    endpoint: 'https://api.twitter.com/1.1/account/verify_credentials.json'
oauth_signature_method: 'HMAC-SHA1'
field_mapping:
    email: email
    first_name: name
    last_name: name
    username: screen_name
```

- Windows Live

```
authorization_endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/authorize'
token_endpoint:
  endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/token'
scope: openid offline_access profile email
field_mapping:
  email: email
  username: preferred_username
  first_name: name
  last_name: name
```

- Standard

```
response_type: code
scope: openid
field_mapping:
  username: sub
  email: email
  last_name: family_name
  first_name: given_name
credential_location: auth_header
```

  Although this is the default configuration for the **standard** provider type, you should contact your OIDC provider to obtain the details of the fields that you need to define.

Creating your OIDC user registry

To create your OIDC user registry, use the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org admin oidc_config_file
```

where:

- *mgmt_endpoint_url* is the platform API endpoint URL.
- *organization_name* is the value of the **name** property of your provider organization.
- *oidc_config_file* is the name of the YAML file that defines the configuration of your OIDC user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

By default, the *registry_name* is derived from the **title** property in the configuration YAML file but you can override this by including a **name** property in the file. The *registry_url* is an internal URL that API Connect assigns to the registry.
After you have created your OIDC user registry, you must update your application registration with the third party OIDC provider to include the OIDC redirect URI; you can obtain this information by using the following command, which displays the details of the registry in the command window:

```
apic user-registries:get --server mgmt_endpoint_url --org organization_name registry_name --output -
```

The required **oidc_redirect_uri** value is in the **consumer:** section; for example:

```
consumer:
  oidc_redirect_uri: https://consumer.mycompany.com/consumer-api/oauth2/redirect
```

Enabling your OIDC registry in a Catalog

If you want to make your OIDC registry available for onboarding and authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your OIDC user registry by using the following command:

   ```
   apic user-registries:list --server mgmt_endpoint_url --org admin
   ```

2. Log in to the Management server as a member of a provider organization; enter the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   For full details of the **apic login** command, see Logging in to a management server.
3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
   ```

   where *catalog_name* is the value of the `name` property of the required Catalog. The command returns

   ```
   Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
   ```

4. Enter the following data, followed by a new line:

   ```
   user_registry_url: oidc_registry_url
   ```

   where *oidc_registry_url* is the URL of your OIDC registry, obtained in step 1.
5. Press **CTRL D** to terminate the input.

For details of all the **apic user-registries** and **apic configured-catalog-user-registries** commands, see apic user-registries and apic configured-catalog-user-registries.

You can also complete the operations described in this topic by using the API Connect REST APIs; see the API Connect REST API documentation.

Setting visibility for a user registry

The visibility setting determines which provider organizations can access a resource, for example, user registries.

Before you begin

You must complete the following tasks:

- Creating an availability zone
- Registering a gateway service

About this task

The user registry visibility setting controls which provider organizations are able to use that registry for authenticating Developer Portal users, or for securing APIs. For details on how to specify the user registries that can be used for authenticating users of the Cloud Manager and API Manager user interfaces, see Selecting user registries for Cloud Manager and API Manager.

Procedure

Follow these steps to set the visibility for the resources (user registries) in your on-premises cloud:

1. In the Cloud Manager, click Resources.
2. Select User Registries as the resource to view.
3. From the list of user registries, choose Edit visibility from the actions menu next to the name of the user registry you are working with.
4. Select the visibility setting for the user registry. The options are:
   - Private - the user registry is not visible and cannot be used by any provider organization
   - Public - the user registry is visible and can be used by all provider organizations
   - Custom - the user registry is visible only to the provider organizations designated by you
5. For Custom visibility, select the provider organizations that you want to have access to the user registry.
6. Click Make visible to complete the operation.

Results

For Private, the user registry cannot be used by any provider organizations. For Public, the user registry can be used by all provider organizations. For Custom, the user registry can be used by the provider organizations that you designate.

Deleting a user registry

As the Cloud Manager administrator, you can delete user registries.

About this task

Complete the following steps to delete a user registry. All users within the user registry will be deleted. The pre-configured user registries (API Manager Local User Registry and Cloud Manager Local User Registry) may not be deleted. Any user registry marked as the Default in Settings > User Registries also may not be deleted.

One of the following roles is required to delete a user registry:

- Administrator
- Owner

Procedure

1. In the Cloud Manager, click .
2. Select User Registries to see the list of current user registries in your cloud.
3. Choose Delete from the overflow menu that is adjacent to the user registry you want to delete.
4. Optionally, you can delete multiple user registries by marking the checkboxes and selecting Delete Selected User Registries from the overflow menu for the list.
5. Confirm the deletion.

Results

The user registry is deleted and removed from the User Registries list in Cloud Manager. All user accounts contained in the user registry are deleted.

Removing a user from a user registry

As an API Connect user, you can remove yourself from an API Connect user registry. As an administrative user, you can remove other users from an API Connect user registry.

For details on how to remove a user from a user registry, see the following subtopics:

- **Removing yourself from a user registry**
  As an API Connect user, you can remove yourself from the API Connect user registry in which you were registered, by using the developer toolkit CLI.
- **Removing another user from a user registry**
  You can remove a user from an API Connect user registry by using the developer toolkit CLI.

Removing yourself from a user registry

As an API Connect user, you can remove yourself from the API Connect user registry in which you were registered, by using the developer toolkit CLI.

Before you begin

You cannot remove yourself from a user registry if you are a member of a provider organization. Before removing yourself from a user registry, ensure that you have been deleted from any provider organizations. For more information, see Removing a user from an organization.

Procedure

1. Log in to the management server as the user that you want to remove.
   The format of the login command depends on whether you are a user in the Cloud Manager admin organization, or you are a user in a provider organization.

   If you are a Cloud Manager admin user, enter the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
   ```

   If you are a provider organization user, enter the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   If you want to log in as a member of the cloud administration organization, or as a member of a provider organization, you can help determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope scope --server mgmt_endpoint_url --fields title,realm
   ```

   where *scope* has the value `admin` or `provider` depending on whether you want to log in as a member of the cloud administration organization, or as a member of a provider organization. The output lists the names and titles of all identity providers, for example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`, and the default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

   Note: If the same user registry is used for both the Cloud Manager and API Manager user interfaces, and you have access to both, when you remove yourself from the user registry you will lose access to both user interfaces regardless of which organization you log in to.
2. Remove yourself from the user registry. Enter the following command:

   ```
   apic me:delete --server mgmt_endpoint_url
   ```

   For example:

   ```
   apic me:delete --server platform-api.myserver.com.com
   ```

   The command confirms successful removal by returning the details of the deleted user; for example:

   ```
   user1    [state: enabled]   https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-
   0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
   ```

Removing another user from a user registry

You can remove a user from an API Connect user registry by using the developer toolkit CLI.

Before you begin

To complete this task, you must have Administrator access to your API Connect cloud, or be the owner of a provider organization, depending on the scope of the user registry.

You cannot remove a user from a user registry if that user is a member of the Cloud Manager admin organization, a provider organization, or a consumer organization, and that user registry is used for the associated login. Before removing a user from a user registry, ensure that they have been deleted from any organizations for which that user registry is used for login. For more information, see Deleting a member or Removing a user from an organization.

Procedure

1. Log in to the management server.
   - If the user registry is defined at the cloud administration level, and therefore available to all provider organizations, you must log in as a member of the Cloud Manager admin organization.
   - If the user registry is defined at a provider organization level, and therefore specific to that provider organization, you must log in as the organization owner.

   For full details of the **apic login** command, see Logging in to a management server.

   Note: If the same user registry is used for both the Cloud Manager and API Manager user interfaces, and the user has access to both, when you remove them from the user registry they will lose access to both even though you are logging in to the admin organization.
2. Identify the name of the user registry from which you want to remove the user. Enter the following command:

   ```
   apic user-registries:list --server mgmt_endpoint_url --org organization_name
   ```

   where *organization_name* has the value **admin** if you are logged in as a member of the Cloud Manager admin organization, or is the name of the provider organization if you are logged in as an organization owner.
   For example:

   ```
   apic user-registries:list --server platform-api.myserver.com.com --org admin
   ```

   The command returns a list of all user registries, with the registry name displayed first; for example:

   ```
   api-manager-lur     https://platform-api.myserver.com.com/api/user-registries/3283e897-1d23-4fac-acf5-
   0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245
   cloud-manager-lur   https://platform-api.myserver.com/api/user-registries/3283e897-1d23-4fac-acf5-0193d0b2c1b5/3adbf524-
   cd74-4051-99c0-89ce5ffcc9c0
   my-ldap             https://platform-api.myserver.com/api/user-registries/3283e897-1d23-4fac-acf5-0193d0b2c1b5/29eae413-
   cd74-4051-99c0-89ce5ffcc9c0
   ```

3. Identify the name of the user, from the required user registry, that you want to remove. Enter the following command:

   ```
   apic users:list --server mgmt_endpoint_url --org organization_name --user-registry user_registry_name
   ```

   For example:

   ```
   apic users:list --server platform-api.myserver.com.com --org admin --user-registry my-ldap
   ```

   The command returns a list of all users in the user registry, with the user name displayed first; for example:

   ```
   user1     [state: enabled]   https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-
   0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
   user2     [state: enabled]   https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-
   0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/12f1aa82-d9f0-4670-99f1-7500d1fb6583
   ```

4. Remove the required user from the user registry. Enter the following command:

   ```
   apic users:delete user_name --server mgmt_endpoint_url --org organization_name --user-registry user_registry_name
   ```

   For example:

   ```
   apic users:delete user1 --server platform-api.myserver.com.com --org admin --user-registry my-ldap
   ```

   The command confirms successful removal by returning the details of the deleted user; for example:

   ```
   user1    [state: enabled]   https://platform-api.myserver.com.com/api/user-registries/32830897-1d23-4fac-acf5-
   0193d0b2c1b5/4438937a-6ad0-4eaa-9163-820888ac6245/users/040adb11-e9a4-4d93-9c2e-62a974da0689
   ```

TLS profiles overview

API Connect supports TLS Profiles for securing data transmission over HTTPS.

Introduction to TLS profiles

Important: API Connect includes several default TLS profiles to help you get started working with the application. The default profiles should not be used in a production environment. It is important to create your own profiles to ensure a secure network.
API Connect may need to transmit data across an untrusted network, for example, when accessing the Gateway, email server, or LDAP server. TLS provides secure network layer transportation of data between two parties.

There are two types of TLS Profiles: a TLS Server Profile and a TLS Client Profile. A TLS Server Profile is used by the Gateway to configure its endpoint for use during API execution. A TLS Client profile is used whenever the system needs to communicate with another endpoint over TLS.

The components of a TLS Profile are:

- TLS Protocol version indicates the versions of the Transport Layer Security Protocol required for the profile. TLS Protocol versions 1.0, 1.1, 1.2, and 1.3 are supported.
- Optional support for mutual authentication and renegotiation for Server Profiles.
- Optional support for weak server connections and Server Name Indication for Client Profiles.
- Cipher suites to secure HTTPs communication within the API Connect ecosystem.
- Keystores containing public and private key pairs.
- Truststores containing public keys for trusted third party services, such as Google, Facebook, or Verisign.

Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.

- **Viewing TLS Profiles, Keystores, and Truststores**
  The current list of TLS profiles, Keystores, and Truststores can be viewed on the main TLS screen.
- **Creating a TLS Server Profile**
  In Cloud Manager, you can configure the profile that is used on the gateway when it acts as a TLS server.
- **Creating a TLS Client Profile**
  In Cloud Manager, TLS profiles are configured as a Resource to provide secure transmission of data over HTTPs.
- **Creating a Keystore**
  Each keystore contain a matched pair of a public certificates and its private keys. These artifacts provide identity information during a TLS handshake.
- **Creating a Truststore**
  A truststore contains a list of certificates. The certificates are used to verify the peer during a TLS handshake.
- **Viewing certificate details and adding certificates to a keystore or truststore**
  You can view details for the certificates in an existing keystore or truststore and add additional certificates.
- **Generating a self-signed certificate using OpenSSL**
  OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.
- **Generating a PKCS#12 file for Certificate Authority**
  PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.
- **Setting the API invocation certificate for a gateway service**
  Configure the TLS certificate for the API invocation endpoint on your gateway service.
- **Updating the PKCS#12 certificate for a TLS server profile**
  A server certificate bound to a gateway service can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client, or because it has expired. When this happens, you must update the TLS profile with a new CA certificate or PKCS#12 (P12) file.

Related tasks

- Creating a TLS Server Profile
- Creating a TLS Client Profile
- Creating a Keystore
- Creating a Truststore

Viewing TLS Profiles, Keystores, and Truststores

The current list of TLS profiles, Keystores, and Truststores can be viewed on the main TLS screen.

Before you begin

Important: API Connect includes several default TLS profiles to help you get started working with the application. The default profiles should not be used in a production environment. It is important to create your own profiles to ensure a secure network.
One of the following roles is required to view TLS Server Profiles:

- Administrator
- Owner
- Topology Administrator
- Viewer
- Custom role with the `Settings: Manage` or `Settings: view` permissions

About this task

The TLS list screen provides a list of TLS Profiles, Keystores and Truststores configured for your cloud. Users with view permissions can use this screen as a reference for the TLS Profiles.

Users with manage permissions can create new Profiles, Keystores, and Truststores; edit existing ones; and delete items that are no longer needed. Users can view the Keystores and Truststores assigned to the profiles.

Users can view and set the visibility for Client Profiles. For more information on visibility, see Setting visibility for a TLS Client Profile

Procedure

Following are the tasks you can perform using the list screen as a base:

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Complete your management task:
   - To create a new item, click Create next to the table for the type of item you want to create.
   - To edit an existing item, either click the Title of the item, or click the options menu ⋮ and then click Edit.
   - To set the visibility of a TLS Client Profile, click the options menu ⋮ for the profile, and then click Edit Visibility.
   - To delete an existing item, click the options menu ⋮ for the profile, and then click Delete.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Server Profile
- Creating a TLS Client Profile
- Creating a Keystore
- Creating a Truststore

Creating a TLS Server Profile

In Cloud Manager, you can configure the profile that is used on the gateway when it acts as a TLS server.

Before you begin

Important: API Connect includes several default TLS profiles to help you get started. The default profiles should not be used in a production environment. It is important to create your own profiles to secure your network.
One of the following roles is required to configure TLS Server Profiles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

About this task

The Server profile is for the gateway when it is acting as the TLS server.

Procedure

Perform the following steps to create a TLS Server profile:

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Click Create in the TLS Server Profile table.
4. Enter the fields to configure the TLS Server Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage TLS Server Profiles, see apic tls-server-profiles.
Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, *MyProfile 1.0* and *MyProfile 1.1*.
Summary (optional)	Enter a description of the profile.
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Mutual Authentication (required)	Determines the level of two-way authentication for the server profile. In two-way authentication, the server responds to a client by sending a request for the client certificate. • None (default) No support for mutual authentication. • Request Enable this option to request client authentication during the TLS handshake. When the application sends the request, the gateway requests that the application sends the certificate. If the client does not send the certificate, the certificate is not checked on the gateway. • Require Enable this option to require client authentication during the TLS handshake. When the application sends the request, the gateway requests that the application sends the certificate. If the client does not send the certificate, the TLS handshake fails and the request is blocked.
Limit Renegotiation (optional)	Client-initiated renegotiation allows the connection to be retried. The default is to prevent renegotiation. Remove the checkmark to allow renegotiation.
Keystore (required)	A keystore is a repository containing a public and private key pair. The Server Profile requires a keystore in order to securely identify the system. When an application sends an API request, the keystore is used to verify a matching certificate. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.
Truststore (optional)	A truststore is a repository containing certificates. The certificates are used to verify the peer during a TLS handshake. If, in addition to a keystore, a truststore is specified, the certificate is further checked for validity by ensuring that is signed by the root certificate, which must be in the truststore. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.
Ciphers (required)	Cipher suites are encryption/decryption algorithms used to secure HTTPs communication within the API Connect ecosystem. Select the ciphers that the profile supports. Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, **all** the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are **not** added to the list of ciphers supported by the profile.

5. Click Save.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Client Profile
- Creating a Keystore
- Creating a Truststore

Creating a TLS Client Profile

In Cloud Manager, TLS profiles are configured as a Resource to provide secure transmission of data over HTTPs.

Before you begin

One of the following roles is required to configure TLS Profiles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

API Connect uses both TLS Server and TLS Client profiles. A TLS Server profile is presented when a communication request is received. The Server profile validates the request against the Keystore and protocol version to determine whether the connection is secure. The Client profile is presented to initiate communication with another system. Client Profiles may be made visible for use by provider organizations in policies in API Manager. See Setting visibility for a TLS Client Profile.

Procedure

Perform the following steps to create a TLS Client profile:

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Click Create in the TLS Client Profile table.
4. Enter the fields to configure the TLS Client Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage a TLS Client Profile, see apic tls-client-profiles.
Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, *MyProfile 1.0* and *MyProfile 1.1*.
Summary (optional)	Enter a description of the profile.
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Server Connection (optional)	Specify whether to support weak or insecure credentials. • Allow insecure server connections - Insecure server connections may result from self-signed certificates, expired or corrupted certificates, or certificates from an unknown or untrusted source. Check this box to allow the connection to proceed with an insecure connection. The default is to not allow insecure server connections. • Support Server Name Indication (SNI) - Check this box to enable SNI. SNI allows support for multiple certificates presented on the same IP address using different host names. The client profile sends the name of a virtual domain as part of the TLS negotiation. The default is to enable SNI.
Keystore (optional)	A Keystore is a repository containing public and private key pairs. Select the keystore where you will store the certificates for the profile. Default keystores are provided, and you can also create your own. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your uploaded certificates before they expire.
Truststore (optional)	A Truststore is a repository containing verified public keys, which are usually obtained from a third-party certificate authority. Truststores provide list of certificates to verify a peer's certificate. If used in a TLSServerProfile, a truststore is used when mutual authentication is enabled. Select a truststore for the profile. Default truststores are provided, and you can also create your own. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your uploaded certificates before they expire.
Ciphers (required)	Cipher suites are encryption/decryption algorithms used to secure HTTPs communication within the API Connect ecosystem. Select the ciphers that the profile supports. Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, **all** the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are **not** added to the list of ciphers supported by the profile.

5. Click Save.

- **Setting visibility for a TLS Client Profile**
  The visibility setting determines which provider organizations can access a resource.
- **Defining elliptic curve cryptographic schemes for a TLS client profile**
  You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

Related concepts

- TLS profiles overview

Related tasks

- Setting visibility for a TLS Client Profile
- Creating a TLS Server Profile
- Creating a Keystore
- Creating a Truststore

Setting visibility for a TLS Client Profile

The visibility setting determines which provider organizations can access a resource.

Before you begin

You must complete the following task:

- Creating a TLS Client Profile

About this task

The visibility setting gives you control over the resources that are available to provider organizations.

Procedure

Follow these steps to set the visibility for the resources in your on-premises cloud:

1. In the Cloud Manager, click Resources.
2. Select TLS as the resource to view.
3. From the table of TLS Client Profiles, choose Edit visibility from the ⋮ options menu next to the name of the profile you are managing.
4. Select the visibility setting for the profile. The options are:
   - Private - the profile is visible only in Cloud Manager and cannot be used by any provider organization
   - Public - the profile is visible and can be used by all provider organizations
   - Custom - the profile is visible only to the provider organizations designated by you
5. For Custom visibility, select the provider organizations that you want to have access to the profile.
6. Click Save to complete the operation.

Results

For Private, the profile cannot be used by any provider organizations. For Public, the profile can be used by all provider organizations. For Custom, the profile can be used by the provider organizations that you designate. If visible, the profiles will be available in API Manager.

Defining elliptic curve cryptographic schemes for a TLS client profile

You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

About this task

To define elliptic curve cryptographic schemes for a TLS client profile, you include **`elliptic_curve_auto_negotiation`** and **`elliptic_curve`** properties in a YAML file definition for the TLS client profile. The **`elliptic_curve`** property lists the required elliptic curve cryptographic schemes. For example:

```
elliptic_curve_auto_negotiation: false
elliptic_curve:
  - secp521r1
  - secp384r1
  - prime256v1
```

You then use the developer toolkit CLI to create the TLS client profile in API Connect.

When **`elliptic_curve_auto_negotiation`** is set to **`true`**, the system negotiates the Elliptic-curve Diffie-Hellman (ECDH) key agreement automatically with its peer, and any **`elliptic_curve`** property settings are ignored.

The following example shows a complete YAML file for a TLS client profile:

```
type: tls_client_profile
name: my-tls-client-profile
version: 1.0.0
title: My TLS client profile
protocols:
  - tls_v1.2
ciphers:
  - ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
elliptic_curve_auto_negotiation: false
elliptic_curve:
  - sect163k1
insecure_server_connections: false
server_name_indication: true
```

Note:

- The **elliptic_curve_auto_negotiation** option is not supported by any of the API Connect gateway types. If the TLS client profile is targeted for an API Connect gateway; this setting is ignored by the gateway.
- The elliptic curve cryptographic schemes shown in each row of the following table are equivalent. However, API Connect recognizes only one or the other depending on how the TLS profile is used, as indicated in the table.

  Table 1.

API enforcement on the gateway	API Connect server access security
secp192r1	prime192v1
secp256r1	prime256v1

  Therefore, if you want to use either of these schemes and are unsure whether you are targeting the TLS client profile to the API Connect gateway for API enforcement, whether you are using it to secure user access to the API Connect servers, or whether it will be used for both purposes, specify both equivalent schemes; API Connect will simply ignore the non-relevant scheme. For example:

```
elliptic_curve:
        .
        .
        .
  - secp256r1
  - prime256v1
        .
        .
        .
```

Procedure

To create a TLS client profile with elliptic curve cryptographic schemes defined. complete the following steps:

1. Create a YAML file definition for your TLS client profile, with the required **elliptic_curve** property.
2. Log in to the management server from the developer toolkit CLI. Log in either as a member of the cloud administration organization or as a member of a provider organization, depending on where you want to create the TLS client profile. For details, see Logging in to a management server.
3. Create the TLS client profile by using the following command:

   ```
   apic tls-client-profiles:create --server mgmt_endpoint_url --org organization_name tls_client_profile_yaml_file
   ```

   where:
   - *mgmt_endpoint_url* is the platform API endpoint URL, and is the same as that which was used when you logged in at step 2.
   - *organization_name* is either **admin**, for the cloud administration organization, or the name of your provider organization, and is the same as that which was used when you logged in at step 2.
   - *tls_client_profile_yaml_file* is the name of the YAML file that contains the definition for your TLS client profile.

   Note: When you install IBM® API Connect, the API Connect gateway has a pre-supplied default TLS client profile that is used for API enforcement if you do not configure a TLS client profile; you cannot configure this default TLS client profile on the gateway.
   For reference details of all the **apic tls-client-profiles** commands, see apic tls-client-profiles.

   You can also complete the operations described in this topic by using the API Connect REST APIs; see the API Connect REST API documentation.

Creating a Keystore

Each keystore contain a matched pair of a public certificates and its private keys. These artifacts provide identity information during a TLS handshake.

Before you begin

Cloud Manager and API Manager both support and use TLS certificates, but they do not themselves produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see Generating a PKCS#12 file for Certificate Authority and Generating a self-signed certificate using OpenSSL.

One of the following roles is required to configure Keystores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

API Connect includes pre-configured Keystores which may be used for testing purposes. For production environments, we suggest creating a new, secure Keystore.

Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.

Procedure

Perform the following steps to create a TLS Client profile:

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Click Create in the Keystore table.

Field	Description
Title (required)	Enter a Title for the Keystore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage keystores, see apic keystores.
Summary (optional)	Enter a brief description.
Private Key & Public Key: Step 1: Upload private key	Upload the file containing the private key certificate. If necessary, you can click Browse to locate the file. If the file contains both the private and public keys, upload it in Step 1. Private and public keys are always uploaded in pairs, either in a single file or separate files.
Private key password (optional)	Enter the password for the private key if it has a password.
Private Key & Public Key: Step 2: Upload public key	If the public key is contained in a separate file, upload it in Step 2. Private and Public keys are always uploaded in pairs, either in a single file or separate files.

4. Click Save.
   Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Client Profile
- Creating a Truststore
- Generating a PKCS#12 file for Certificate Authority
- Generating a self-signed certificate using OpenSSL

Creating a Truststore

A truststore contains a list of certificates. The certificates are used to verify the peer during a TLS handshake.

Before you begin

One of the following roles is required to configure Truststores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

Cloud Manager and API Manager both support and use TLS certificates, but they do not themselves produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see Generating a PKCS#12 file for Certificate Authority and Generating a self-signed certificate using OpenSSL.

API Connect includes pre-configured Truststores which may be used for testing purposes. For production environments, we suggest creating a new, secure Truststore.

Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.

Procedure

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Click Create in the Truststore table.

Field	
Title (required)	Enter a Title for the Truststore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage truststores, see apic truststores.
Summary (optional)	Enter a brief description.
Public Keys	Upload the file containing the public key certificate. If necessary you can click Browse to locate the file.

4. Click Save.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Client Profile
- Creating a Keystore

Viewing certificate details and adding certificates to a keystore or truststore

You can view details for the certificates in an existing keystore or truststore and add additional certificates.

Before you begin

One of the following roles is required to edit Keystores and Truststores:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

API Connect includes pre-configured Keystores and Truststores which you can use for testing purposes. For production environments, we suggest creating a new, secure Keystore or Truststore. From an existing Keystore or Truststore, you can edit the title, view the details of the certificates, and add new certificates.
Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.

Procedure

To work with existing keystores and truststores:

1. In the Cloud Manager, click Resources.
2. Select TLS.
3. Click the name of the keystore or truststore.
4. Edit the name and summary if needed.
5. The Subject, Finger Print, and Expiration date is shown for each certificate in the keystore or truststore. Click > to view the certificate details.
6. Add additional private and/or public keys if needed.
7. Click Save.
   Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Server Profile
- Creating a TLS Client Profile
- Creating a Truststore
- Creating a Keystore
- Generating a PKCS#12 file for Certificate Authority
- Generating a self-signed certificate using OpenSSL

Generating a self-signed certificate using OpenSSL

OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.

Before you begin

One of the following roles is required to complete this task:

- Administrator
- Owner
- Topology Administrator

- Custom role with the `Settings: Manage` permissions

About this task

This topic tells you how to generate a self-signed SSL certificate request using the OpenSSL toolkit to enable HTTPS connections.

Procedure

To generate a self-signed SSL certificate using the OpenSSL, complete the following steps:

1. Write down the Common Name (CN) for your SSL Certificate. The CN is the fully qualified name for the system that uses the certificate. For static DNS, use the hostname or IP address set in your Gateway Cluster (for example. **`192.16.183.131`** or **`dp1.acme.com`**).
2. Run the following OpenSSL command to generate your private key and public certificate. Answer the questions and enter the Common Name when prompted.

   ```
   openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
   ```

3. Review the created certificate:

   ```
   openssl x509 -text -noout -in certificate.pem
   ```

4. Combine your key and certificate in a PKCS#12 (P12) bundle:

   ```
   openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12
   ```

5. Validate your P2 file.

   ```
   openssl pkcs12 -in certificate.p12 -noout -info
   ```

   Once the certificate file is created, it can be uploaded to a keystore.
6. In the Cloud Manager, click Resources.
7. Select TLS.
8. Click Create in the Keystore table.
9. Create a Keystore and upload the certificate file following the instructions at Creating a Keystore.
   Note:
   - API Connect supports only the P12 (PKCS12) format file for the present certificate.
   - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
   - Your P12 file can contain a maximum of 10 intermediate certificates.
10. Click Save.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Server Profile
- Creating a TLS Client Profile
- Creating a Keystore
- Creating a Truststore

Generating a PKCS#12 file for Certificate Authority

PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.

Before you begin

One of the following roles is required to add a key to a keystore or truststore:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

Before you can generate a P12 file, you must have a private key (for example: *key.pem*), a signed certificate by a Certificate Authority (for example *certificate.pem*) and one or more certificates from the CA authority.
Note: If your certificate file contains more than one certificate, you must manually split the file and create a single file for each entry. Each entry must be bound by the following markers:

```
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-------
```

Procedure

1. If you have intermediate certificates from your CA, concatenate them into a single .pem file to build your **`caChain`**. Be sure to enter a new line following each certificate's data.

```
cat ca1.pem ca2.pem ca3.pem > caChain.pem
cat caChain.pem
-----BEGIN CERTIFICATE-----
MIIEpjCCA46gAwIBAgIQEOd26KZabjd+BQMG1Dwl6jANBgkqhkiG9w0BAQUFADCB
...
lQX7CkTJn6lAJUsyEa8H/gjVQnHp4VOLFR/dKgeVcCRvZF7Tt5AuiyHY
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEPDCCAySgAwIBAgIQSEus8arH1xND0aJ0NUmXJTANBgkqhkiG9w0BAQUFADBv
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIENjCCAx6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEwJTRTEU
...
-----END CERTIFICATE-----
```

2. Create the P12 file including the private key, the signed certificate and the CA file you created in step 1, if applicable. Omit the `-CAfile` option if you don't have CA certificates to include.
   The following command uses OpenSSL, an open source implementation of the SSL and TLS protocols.

   ```
   openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12 -CAfile caChain.pem -chain
   ```

   Once the certificate file is created, it can be uploaded to a keystore.
3. In the Cloud Manager, click Resources.
4. Select TLS.
5. Click Create in the Keystore table.
6. Create a Keystore and upload the certificate file following the instructions at Creating a Keystore.
   Notes:
   - API Connect supports only the P12 (PKCS12) format file for the present certificate.
   - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
   - Your P12 file can contain a maximum of 10 intermediate certificates.
7. Click Save.

Related concepts

- TLS profiles overview

Related tasks

- Creating a TLS Server Profile
- Creating a TLS Client Profile
- Creating a Keystore
- Creating a Truststore
- Generating a self-signed certificate using OpenSSL

Setting the API invocation certificate for a gateway service

Configure the TLS certificate for the API invocation endpoint on your gateway service.

Before you begin

One of the following roles is required:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Topology: Manage` permissions

For more information on which user roles have access, see Adding members to the admin organization.

About this task

To set the API invocation certificate for a gateway service, you must add the certificate to a TLS server profile, and then update your gateway service to use this TLS server profile.

Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your certificates before they expire.

Procedure

1. Create or update the TLS server profile that contains the certificate you want for your API invocation endpoint. See Creating a TLS Server Profile.
2. In the Cloud Manager user interface, click Topology, then select the gateway service where you want to set the API invocation certificate.
3. Scroll down to the API Invocation Endpoint section, and under Server Name Indication (SNI) select the TLS server profile that you configured in step 1.
   You can also set the TLS server profile during the initial registration of a gateway service; see Registering a gateway service.
4. Click Save when done.

Updating the PKCS#12 certificate for a TLS server profile

A server certificate bound to a gateway service can be invalidated if the host name in the digital certificate of the server does not match the URL specified by the client, or because it has expired. When this happens, you must update the TLS profile with a new CA certificate or PKCS#12 (P12) file.

Before you begin

One of the following roles is required:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

If the expiration date of a certificate or a P12 file is approaching, or if a certificate is invalidated, use the steps in this topic to update a TLS profile bound to a gateway service. CA certificate and P12 file expiration dates are displayed in the details of the containing keystore; see step 3.

You update the certificate for a TLS server profile by replacing the certificate in the keystore that is associated with the TLS server profile.

Complete the following steps to update a TLS profile that has an invalidated or expired certificate or P12 file.

Procedure

1. In the Cloud Manager user interface, click Resources, then click TLS.
2. To identify the keystore that is associated with the TLS server profile, complete the following steps:
   a. In the TLS Server Profile section, select the required profile.
   b. In the Keystore/Truststore section, note the selected keystore, then click Cancel to close the TLS server profile details page.
3. In the Keystore section, select the required keystore.
   The Certificates section displays the expiration date of a certificate. You can expand a certificate to see further details.
4. Click Browse and select the required P12 file.
   Note:
   - API Connect supports only the P12 (PKCS12) format file for the present certificate.
   - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
   - Your P12 file can contain a maximum of 10 intermediate certificates.
5. Click Save when done.

Related tasks

- Creating a TLS Server Profile

OAuth Provider overview

API Connect supports OAuth Specification 2.0, for both Native and Third party implementations.

Introduction to OAuth

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information. In API Connect, you can secure an API with OAuth.

In Cloud Manager, you configure both Native and Third party OAuth providers that can be made visible to selected Provider organizations. The OAuth Provider configuration is based on the OAuth 2.0 Specification, which is available at https://tools.ietf.org/html/rfc6749. Knowledge of the OAuth 2.0 specification is required to implement an OAuth Provider in API Connect.

One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- **Configuring a native OAuth provider**
  Native OAuth providers are configured and managed by you within your cloud.
- **Configuring a third-party OAuth provider**
  Enter the secure endpoints to provide OAuth authentication from a third party.
- **Setting the visibility for OAuth providers**
  The visibility setting determines which provider organizations can use an OAuth provider to secure APIs.
- **OAuth concepts for API Connect**

Configuring a native OAuth provider

Native OAuth providers are configured and managed by you within your cloud.

About this task

A native OAuth provider object provides settings for OAuth processing operations such as generating and validating OAuth tokens. An OAuth provider object is referenced by an OAuth security definition to protect an API. When a native OAuth provider is used, the OAuth operations are performed natively by API Connect.

Every OAuth provider object has a backing API. Your configuration here automatically updates the OpenAPI document of the API. You can edit the OpenAPI document directly by navigating to the Resources > OAuth Providers page, selecting your OAuth provider, then clicking API Editor.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the **`oauth-provider-settings-ref`** field in each OAuth assembly action.

When a published API references an OAuth provider object, the backing API is automatically made available in the gateway.

Token requests and client_id checks:
If you are calling an OAuth endpoint, the sequence of **`client_id`** checks in the token request is as follows:

1. Check both body and query.
   - If found in only the body, validate and return a **`200`** or appropriate return code.
   - If found in only the query, return a **`wrong location`** error.
   - If found in both the body and the query, return a **`more than one location`** error.
2. When not found in the body or the query, check the **`Authorization`** header.
   - If found in this header, validate and return a **`200`** or appropriate return code.
3. Not found, return the appropriate error and code.

The best location for the **`client_id`** is in the request body.
One of the following roles is required to configure a native OAuth provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note:

- The OAuth provider logs analytics data for failure cases, but does not log successful cases. Activity log policies that call for logging of analytics data upon success do not apply for the OAuth provider.
- You must ensure the OAuth Provider is configured in the Sandbox Catalog before using the OAuth Provider in a non-Sandbox Catalog.

Procedure

OAuth provider configuration uses a series of screens. The first set of screens controls a basic OAuth provider configuration. Perform the following steps to configure a native OAuth Provider for your cloud:

1. In the Cloud Manager, click Resources.
2. Click OAuth Providers > Add > Native OAuth provider.
   a. Complete the following parameters for the first screen, then click Next.

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
Gateway Type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see API Connect gateway types. OAuth Providers apply to one gateway type.

   b. In the next screen, enter the following additional configuration parameters, then click Next.

Field	Description
Authorize Path	`/oauth2/authorize/` is the standard OAuth endpoint to login to account
Token Path	`/oauth2/token/` is the standard OAuth endpoint to exchange code for access token.

Field	Description

Supported grant types	• Implicit - An access token is returned immediately without an extra authorization code exchange step.
	• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.
	• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."
	• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.
	• API Gateway only Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.
	Note: To use the Resource owner - JWT option, complete the following steps:
	  • In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT.
	  • Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type.
	  For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).

Supported client types	• Confidential - Client can maintain secure credentials on a secure server • Public - Client credentials are not secure.

c. Enter the scopes in the next screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

d. Enter the parameters for User Security in the next screen. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application grant type. Click Next when done.

Field	Description
Identity Extraction	Determines how the user credential is extracted: • Basic Authentication - HTTP basic authentication (requires no additional configuration) • Default HTML Form - Use default login form for user name and password • API Gateway only Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here API Connect context variables Note: DataPower API Gateway only. Not available for DataPower Gateway (v5 compatible). • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML login form for user security. • Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see Authenticating and authorizing through a redirect URL. • API Gateway only Disabled - do not collect the user credential Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint **must** maintain the order of the query parameters before the `state_nonce` query parameter, otherwise the authorization fails.
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry. For a DataPower API Gateway, you have the option to disable authentication with a user registry.
Authorization	The following methods for extracting the user credential are available: • Authenticated - Authorize authenticated users automatically. • Default HTML Form - Use default HTML form to authorize. • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML authorization form for user security. • API Gateway only Disabled - Disable authorization.

3. Review the Summary for the native OAuth provider configuration.
4. Click Back to make changes.
5. Click Finish to save the basic configurations and to proceed to the Advanced Parameters for a native OAuth Provider.
   If you want to specify the native OAuth Provider in your Catalog, see Specifying the OAuth Providers.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

- **Configuring basic settings for a native OAuth provider**
  You can update the identification details and basic configuration settings for a native OAuth provider.
- **Configuring scopes for a native OAuth provider**
  Access tokens contain authorization for specific scopes.
- **Configuring user security for a native OAuth provider**
  Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.
- **Configuring tokens for a native OAuth provider**
  Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.
- **Configuring token management and revocation for a native OAuth provider**
  Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.
- **Configuring introspection for a native OAuth provider**
  Define an introspection path to allow the metadata for an access token to be examined.
- **Configuring metadata for a native OAuth provider**
  Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- **Configuring the OIDC parameters for a native OAuth provider**
  Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.

- **Editing a native OAuth provider by using the API Editor**
  You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

Configuring basic settings for a native OAuth provider

You can update the identification details and basic configuration settings for a native OAuth provider.

About this task

One of the following roles is required to configure the basic settings for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the basis settings pages for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the basic settings for an existing native OAuth provider. If you want to update the basic settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. To modify the identification details, click Info in the sidebar menu, then update the following fields as required:

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.

2. To modify the basic configuration settings, click Configuration in the sidebar menu, then update the following fields as required:

Field	Description
Authorize Path	`/oauth2/authorize/` is the standard OAuth endpoint to login to account
Token Path	`/oauth2/token/` is the standard OAuth endpoint to exchange code for access token.

Supported grant types	• Implicit - An access token is returned immediately without an extra authorization code exchange step.
	• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.
	• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."
	• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.
	• API Gateway only Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.
	Note: To use the Resource owner - JWT option, complete the following steps:
	  • In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT.
	  • Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type.
	  For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).
	
	Tip: If you plan to configure OpenID Connect (OIDC) for a Native OAuth provider, include at least one of the following grant types: Implicit, Access code.

Supported client types	• Confidential - Client can maintain secure credentials on a secure server • Public - Client credentials are not secure.

DataPower Gateway (v5 compatible) only Note: If the gateway type is DataPower® Gateway (v5 compatible) and, when the native OAuth provider was created, only the Application grant type was selected, you cannot add further grant types until you configure the user security settings. In particular, you must specify the user registry for authenticating application users. To configure the user security settings, complete the following steps:
   a. Click User Security in the sidebar menu, then click Edit.
   b. Update the user security settings as required; for more details, see Configuring user security for a native OAuth provider.
   c. Click Save when done.
3. Click Save when done.

Configuring scopes for a native OAuth provider

Access tokens contain authorization for specific scopes.

About this task

Client applications can request only the scopes or a subset of the scopes that you define here. The scopes are included in the access tokens that are generated from the provider. When an OAuth protected API is invoked, the gateway checks the scopes carried in the access tokens against the list of allowed scopes in the security definition for the API to determine whether to grant access.

In addition, you can enforce advanced scope checks. The advanced scope check URLs are invoked after application authentication or after user authentication based on which URLs are configured. The final scope permission that is granted by the access token is the result of all scope checks.

Per IETF RFC 6749, the value of the scope parameter is a list of space-delimited, case-sensitive strings. For more information, see The OAuth 2.0 Authorization Framework.

One of the following roles is required to configure scopes for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the scope settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the scope settings for an existing native OAuth provider. If you want to update the scope settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure the scopes for the access token:

1. Click Scopes in the sidebar menu. The currently configured scopes are listed. Review and update the scopes as required.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

   In the Default scopes section, select the default scopes to be used if the API request doesn't contain any scopes. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.
   If the user authorization method is set to Default HTML Form in the User Security settings, all scopes specified here are added automatically to the authorization consent form.

2. Advanced scope check before token generation. This setting specifies the scope check endpoint where additional scope verification is performed in addition to the basic scopes. The advanced scope check URLs are invoked after application authentication or after owner authentication based on which URLs are configured. The scopes are included in the token and will overwrite any previous scopes.

Field	Description
Application scope check	Allow extra verification by running a scope check from an endpoint. Enter the endpoint and an optional TLS Profile to use for an application scope check.
Owner scope check	Further refine the scope with an additional check. Enter the endpoint and an optional TLS Profile to use for an owner scope check.

   For more information about scope, see Scope
3. Advanced scope check after token generation. This setting specifies an additional scope check at the API consumer level to verify compliance with the scope requirements of the API.

Field	Description
Enabled	Select the check box to enable the advanced scope check after token validation. Enter an optional default validator endpoint.
Use endpoint from API	Select the check box to use the endpoint from the API, or clear the check box to override the endpoint from the API.

   For more information about scope, see Scope
4. Click Save when done.

Results

Depending upon the visibility setting, the OAuth Provider with the specified scopes can be used to secure the APIs in catalog.

Related information

- Scope

Configuring user security for a native OAuth provider

Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.

About this task

User security authenticates the user. It is required for the Implicit, Access code, and Resource owner - Password grant types. It is not used for the Application or Resource owner - JWT grant types.

One of the following roles is required to configure user security for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator

- Custom role with the `Settings:Manage` permissions

You can select the user security settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the user security settings for an existing native OAuth provider. If you want to update the user security settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure the user security settings for the OAuth Provider:

1. Click User Security in the sidebar menu.
2. Specify the following parameters for User Security. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application or Resource owner - JWT grant types. Click Next when done.

Field	Description
Identity Extraction	Determines how the user credential is extracted: • Basic Authentication - HTTP basic authentication (requires no additional configuration) • Default HTML Form - Use default login form for user name and password • API Gateway only Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here API Connect context variables • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML login form for user security. • Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see Authenticating and authorizing through a redirect URL. • API Gateway only Disabled - do not collect the user credential Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint **must** maintain the order of the query parameters before the **`state_nonce`** query parameter, otherwise the authorization fails.
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry.
Authorization	Various methods may be used to authorize application users. For a DataPower® API Gateway, the following methods for extracting the user credential are available: • Authenticated - Authorize authenticated users automatically. • Default HTML Form - Use default HTML form to authorize. If you select the Default HTML Form method, all scopes that are specified in the Scopes settings are added automatically to the authorization consent form. • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. • API Gateway only Disabled - Disable authorization.

3. Click Save when done.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Related information

- Creating a custom HTML login form for user security
- Creating a custom HTML authorization form for user security

Configuring tokens for a native OAuth provider

Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.

About this task

Access tokens are granted to the client application to allow the application to access resources on behalf of the application user. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or more narrow scope. You can also specify how long the consent given by the combination of any number of access and refresh token remains valid.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the token settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the token settings for an existing native OAuth provider. If you want to update the token settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.

2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure tokens for the native OAuth provider:

1. Click Tokens in the sidebar menu.
2. Define the settings to configure tokens.

Field	Description
Access tokens time to live	Enter the expiration time period in seconds for access tokens.
API Gateway only One time use access token	Click the check box to enable one time use for the access token. Access tokens are multiple use by default which allows them to be used for multiple requests. When one time use is enabled, the access token will be consumed after one use. The OAuth flow will need to be repeated to obtain another access token. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: • Configuring token management and revocation for a native OAuth provider (Cloud Manager) • Configuring token management and revocation for a native OAuth provider (API Manager)
Refresh tokens	Click the check box to enable Refresh tokens. Set the Count to limit the number of times a refresh token can be issued. Set the Refresh Token Time to Live value to determine the time to live, or expiration time period, for each refresh token in seconds.
One time use refresh token	Clear the check box to disable one time use for the refresh tokens. Refresh tokens are one time use by default which allows them to be used one time only to generate an access token and a new refresh token. When refresh token one time use is disabled then the refresh token count is limited to one and the refresh token can be used multiple times to generate new access tokens, however, another refresh token will not be generated unless the initial OAuth flow (Authorization Code or Password) is repeated. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: • Configuring token management and revocation for a native OAuth provider (Cloud Manager) • Configuring token management and revocation for a native OAuth provider (API Manager)
Maximum consent	Click the check box to enable Maximum consent and enter the Maximum Consent Time to Live value in seconds. This is the time to live, or expiration time period, for all tokens, both access and refresh.
API Gateway only Token secret	Click the check box to select the Shared Secret which was configured for the gateway. If no Shared Secret was entered in the Gateway Configuration, then enter an key name and key value to use as the token secret.
API Gateway only Proof Key for Code Exchange	Proof Key for Code Exchange (PKCE) is a method to protect OAuth 2.0 public clients from an authorization code interception attack when they use Authorization Code grant requests. You can enable this extension when deploying with the DataPower® API Gateway. For more information, see RFC 7636. Select the options for your OAuth Providers: • Enable proof key for code exchange If selected, enforces PKCE when submitted in Authorization Code grant requests. • Always required If selected, requires PKCE in all Authorization Code grant requests. • Allow plain Select this check box to allow the plain challenge method in Authorization Code grant requests.

3. Click Save when done.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Related information

- Refresh tokens

Configuring token management and revocation for a native OAuth provider

Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.

About this task

Token management enables you to prevent replay attacks by configuring token revocation. API Connect supports token revocation using a native gateway (DataPower) or a third party endpoint. For a native gateway, quota enforcement is used to manage tokens. For a third-party endpoint, a URL to an external service is used to manage tokens.

For more information, see the IETF RFC 7009 OAuth 2.0 Token Revocation.

Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

One of the following roles is required to configure token management and revocation for a native OAuth Provider:

- Administrator

- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the token management settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the token management settings for an existing native OAuth provider. If you want to update the token management settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure revocation settings for tokens:

1. Select Token Management in the sidebar menu.
2. Enable token management by selecting the check box.
3. From the Type list, select either Native or External. Native points to DataPower as the token storage location; External points to a revocation URL for token storage.
   Note: If you are using the API Manager user interface then, for the External option to be available, you must be using DataPower® API Gateway Version 10.0.1.0 or later, and the gateway service must be enabled in the Sandbox Catalog; for details on how to enable a gateway service in a Catalog, see Creating and configuring Catalogs
4. For Native, select one or both of the Resource owner revocation path and Client revocation path.
   - Resource owner revocation path - Uses the standard OAuth revocation path to allow the resource owner (end user) to revoke the application permission.
   - Client revocation path - Uses the standard OAuth revocation path to allow the client (application) to revoke a single token when the application closes.

   For more information about managing tokens with the Native DataPower Gateway, see Token management with the native DataPower Gateway.
5. For External, the settings depend on the gateway type, as follows:
   DataPower API Gateway:
   - Endpoint - the URL of the external management endpoint.
   - TLS Client Profile (optional) - the TLS client profile to secure connections.
   - Security - how to secure connections. The only supported method is basic authentication.
   - Basic authentication username (optional) - the user name for authentication.
   - Basic authentication password (optional) - the password for authentication.
   - Basic authentication request header name (optional) - the request header that contains the authentication string; if you supply both a request header name and user name/password, the request header authentication method is used.
   - Custom header pattern (optional) - the name pattern of the headers to use for sending additional information to the external management service.
   - Cache type - the cache type to control whether and how to cache positive responses. If you select Time to live, specify how long to keep responses in the cache; the default value is 900 seconds.
   - Fail on error - if selected, processing is stopped if the connection to the external management service fails.

   Note: For details of the JSON format that is required when exchanging messages with the external management service, see JSON format to exchange messages with the external management service.
   DataPower Gateway (v5 compatible):
   - Endpoint - Enter the URL to an external web server that contains information about access or refresh tokens. API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing. For more information see Token revocation.
   - TLS Client Profile - Select a TLS profile to verify the external endpoint.
6. Click Save when done.

Results

Depending on the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Configuring introspection for a native OAuth provider

Define an introspection path to allow the metadata for an access token to be examined.

About this task

Token introspection allows an authorized holder of an access token to examine the contents of tokens using an introspection path. The access token to introspect must be obtained through the native OAuth provider. Introspection provides context for the token by allowing an authorized protected resource to query the authorization server to determine the set of metadata for a given token. The metadata includes whether or not the token is currently active, the scopes assigned to the token, and the authorization context in which the token was granted (including who authorized the token and which client it was issued to). API Connect token introspection conforms to IETF RFC 7662. See OAuth 2.0 Token Introspection.

One of the following roles is required to enable introspection for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the introspection settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the introspection settings for an existing native OAuth provider. If you want to update the introspection settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to enable introspection:

1. Click Introspection in the sidebar menu.
2. Select the check box to enable Introspection. The OAuth standard path for introspection, /oauth2/introspect is automatically entered. This path will be used when another entity inspects the token contents.
3. Click Save when done.

Results

Tokens will be queried using the /oauth2/introspect path. Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Configuring metadata for a native OAuth provider

Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

About this task

Configure an Authentication URL or an External URL from which custom metadata is collected for inclusion in the token. The metadata is either stored inside the access token or it is sent along with the access token to the client application. For more information about how the metadata is collected, see OAuth external URL and authentication URL.

Following are examples of metadata that can be included with the access token:

- Metadata about the authenticated resource owner
- Grant type that was used to obtain the token
- A confirmation code to be provided to the client application

One of the following roles is required to configure metadata collection for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can select the metadata settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the metadata settings for an existing native OAuth provider. If you want to update the metadata settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure metadata collection:

1. Click Metadata in the sidebar menu.
2. Select Collect metadata to enable metadata collection.
3. The Authentication URL user registry is selected by default and is required. For more information about the Authentication URL, see Authentication URL user registry.
4. Select the External URL to collect metadata from an external URL. Enter the endpoint and an option TLS Client Profile.
5. API Gateway only If required, override the default Header name token value. The value of this header, if returned in the response from the OAuth endpoint, is placed in the response payload and indicated as **`metadata`**.
6. API Gateway only If required, override the default Header name payload value. The value of this header, if returned in the response from the OAuth endpoint, is placed within the access token and indicated as **`miscinfo`**.
7. Save the OAuth Provider.
8. Click Save when done.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Configuring the OIDC parameters for a native OAuth provider

Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.

About this task

When you enable OpenID connect, a template is provided for generating ID tokens along with access tokens and the required assembly policies are automatically created. You can customize the policies to suit your needs in the API Editor. The sample key provided is for test purposes only and is used to sign the JWT token.

One of the following roles is required to configure an OIDC template for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Note: You can configure OIDC parameters only if the selected grant types for the native OAuth provider include at least one of the Implicit or Access code grant types; see Configuring basic settings for a native OAuth provider.
You can select the OIDC settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the OIDC settings for an existing native OAuth provider. If you want to update the OIDC settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to configure an OIDC template:

1. Click OpenID Connect in the sidebar menu.
2. Select the initial check box to configure an OIDC Template. Enter the following parameters:

Field	Description
API Gateway only Support hybrid response types (optional)	Select the response types for the OpenID Connect hybrid flow to be supported by this OAuth provider.
API Gateway only Auto Generate OIDC API Assembly	Select this option to generate the full OIDC assembly. Leave this option unselected to simply enable OIDC support in the OAuth provider and allow the developer to implement their own assembly.
DataPower Gateway (v5 compatible) only ID token issuer	Descriptive text to indicate the source of the key.
DataPower Gateway (v5 compatible) only ID token signing key	Specify the JSON Web Key (JWK) to be used to sign the ID token.
DataPower Gateway (v5 compatible) only ID token signing algorithm	Select the algorithm used to sign the token.

3. Click Save when done. You can edit the policies by using the API Editor.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Editing a native OAuth provider by using the API Editor

You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

About this task

If you have configured an OIDC template, you can customize it in API Editor. In the API Editor, the Source tab allows you to edit the code for the configuration using a text editor. The API Assemble tab provides a graphical drag-and-drop editor (identical to the one in API Manager) that allows you to add additional elements to the assembly for the OAuth Provider.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the **`oauth-provider-settings-ref`** field in each OAuth assembly action.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

You can modify the native OAuth provider configuration by selecting the API Editor page immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the configuration for an existing native OAuth provider. If you want to update the configuration for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

Perform the following steps to edit the OAuth configuration:

1. Click API Editor in the sidebar menu.
2. In the Source tab, view and edit the policies to customize the behavior for the OAuth provider.
3. In the API Assemble tab, use the drag and drop editor to add additional policies to the OIDC behavior.
   Note: If you add a policy that references a TLS profile, an **invoke** policy for example, then when you publish an API that uses this OAuth provider, you must ensure that the TLS profile is enabled for the Catalog to which you publish the API. For details on how to enable a TLS profile in a Catalog, see Creating and configuring Catalogs.
4. Click Save when done.

Results

Depending upon the visibility setting, the OAuth Provider can be used to secure the APIs in catalog.

Configuring a third-party OAuth provider

Enter the secure endpoints to provide OAuth authentication from a third party.

About this task

One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Procedure

Complete the following steps to configure a third party OAuth provider:

1. In the Cloud Manager, click Resources.
2. Select OAuth Providers > Add > Third party OAuth Provider.
   a. Complete the following parameters for the first screen and click Next.

Field	Description
Title	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name	This field is auto-populated by the system and used as the internal field name.
Supported grant types	Select from the following options: • Implicit: An access token is returned immediately without an extra authorization code exchange step. • Application: Application to application. Corresponds to the OAuth grant type Client Credentials. Does not require User Security. • Access code: An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type Authorization Code. • Resource owner - Password: The user's username and password are exchanged directly for an access token, so can only be used by first-party clients. • Resource owner - JWT: a JSON Web Token (JWT) Bearer Token is used as a means for requesting an OAuth 2.0 access token, and for client authentication, as defined by the JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. Note: To use the Resource owner - JWT option, complete the following steps: ◦ In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT. ◦ Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type. For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).
Gateway type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see API Connect gateway types. OAuth Providers apply to one gateway type.

   b. Specify configuration settings for the endpoints, and then click Next.

Field	Description
Authorization URL	An authorization URL where the resource owner grants authorization to the client application to access a protected resource. Example: `https://example.com/oauth2/authorize`
Token URL	A token request URL where the client application exchanges an authorization grant for an access token. Example: `https://example.com/oauth2/token`
Introspect URL	The introspection URL is where the API gateway validates the access tokens that are issued by the third party provider. Example: `https://example.com/oauth2/introspect` For more information on integrating third party OAuth providers for introspection, see OAuth introspection for third-party OAuth providers.

Field	Description
API Gateway only Introspect cache type	The cache type determines how long responses from the third party provider are cached, if at all. Select one of the following options: • No cache (default): Responses are not cached. • Protocol: Defined by the **`cache-control`** header in the provider response. • Time to live: Defined by the provider.
API Gateway only Cache Time to Live	The length of time, in seconds, for which provider responses are cached, if the Introspect cache type is set to Time to live. The default value is 900.
TLS Profile (optional)	Select an optional TLS profile for communicating with the third party provider.
Security	Default is Basic Authentication.
Basic authentication request header name	The `x-introspect-basic-authorization-header` is available to provide a user-configured HTTP Basic authorization header.
API Gateway only Basic authentication username (optional)	The default user name for HTTP Basic authentication.
API Gateway only Basic authentication password (optional)	The default password for HTTP Basic authentication.
API Gateway only Token validation	Specifies the method used to determine the success of the introspection request that is sent to the third party service to validate the provided token. Select one of the following options: • Connected: The query is successful if the status return code is **`200`**. • Active (default): The query is successful if the status return code is **`200`** and the response JSON body includes the property **`active: true`**.
API Gateway only Custom header pattern (optional)	A regular expression for request headers that are to be passed to the third-party provider; for example, `x-Introspect-*`.
API Gateway only Authorization header pass through	Select this check box if you want to retain the **`Authorization`** header for a bearer token. The default behavior is to remove this header.

c. Enter the scopes in the third screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
sample_scope_2	Scope for token
additional scopes	Scope for token

d. Review the settings on the Summary panel.

3. Click Save and Edit to complete the configuration.
   If you want to specify the third party OAuth Provider in your Catalog, see Specifying the OAuth Providers.

Setting the visibility for OAuth providers

The visibility setting determines which provider organizations can use an OAuth provider to secure APIs.

About this task

One of the following roles is required to configure OAuth Providers:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings:Manage` permissions

Procedure

Follow these steps to set the visibility for the OAuth providers in your on-premises cloud:

1. In the Cloud Manager, click Resources.
2. Select OAuth Providers.
3. Choose Edit visibility from the actions menu next to the name of the OAuth provider that requires a visibility setting.
4. Select the visibility setting for the OAuth provider. The options are:
   - Private - the OAuth provider is not visible and cannot be used by any provider organization. It is accessible only to the admin organization in Cloud Manager.
   - Public - the OAuth provider is visible and can be used by all provider organizations
   - Custom - the OAuth provider is visible only to the provider organizations that you designate
5. For Custom visibility, select the provider organizations that you want to have access to the OAuth provider.
6. Click Save to complete the operation.

Results

For Private, the OAuth provider is not visible and cannot be used by any provider organizations. For Public, the OAuth provider is visible and can be used by all provider organizations. For Custom, the OAuth provider is visible only to the provider organizations designated by you.

OAuth concepts for API Connect

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- **OAuth user scenario**
  Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.
- **OAuth introspection for third-party OAuth providers**
  OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM API Connect can use this feature along with the mentioned provider to protect access to the API.
- **OAuth external URL and authentication URL**
  You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- **Scope**
  Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.
- **Tokens**
  Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.
- **Authentication URL user registry**
  You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.
- **Custom forms for user security**
  You can create custom forms for the authorization and identity extraction phase of OAuth.
- **Securing an API with a JSON Web Token**
  There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM API Connect.
- **Troubleshooting OAuth**
  You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

Related information

- The OAuth 2.0 Framework

OAuth user scenario

Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.

Scenario overview

In this scenario, Alice is a user of an application. Alice can grant permission for an application to access specific information about Alice in a third-party system that is using OAuth. Depending on the type of OAuth that is supported by the target service, Alice does not enter her user name and password into the application. Instead, the application receives an access token that represents her credentials (user name and password). The application can now access the information about Alice in the target system.

For example, Alice maintains a list of books that she reads on a service that is provided by mybooks.com. Following the purchase of a smartphone, Alice installs an application on her new phone to display the book details. The phone application wants to call an API provided by mybooks.com, which can access the information. The mybooks.com API is secured by using the OAuth protocol.

To access the book details, the application must complete a two-step process:

1. The application must first obtain permission from Alice.
2. The application then uses that permission to call the target service and obtain the list of books.

In the first step, the application typically directs Alice to the provider of the target service, mybooks.com. Alice provides her user name and password, and gives permission for the application to access her information. It is important that Alice trusts that she is providing her credentials to the provider of the target service and not to an untrusted proxy application. For example, by checking that the security certificate of the website where Alice enters her credentials matches what Alice expects from the provider of the target service.

The result of this step is the access token that the application can use to call the API. The application then generates the appropriately formatted OAuth request. For example, the Authorization header, or HTTP query parameters, which includes the access token, consumer key, and signature method that are required by OAuth. This OAuth request is used to invoke the API proxy operation.

Scenario within API Connect

No changes to the definition of your API operation are required to support this scenario.

1. Alice grants permission for the application to access her information *before* the invocation of the API.
2. When the application provides the Authorization header, or query parameters, containing the OAuth details about the call to the operation endpoint, the header is automatically passed through to the target service without any additional configuration.

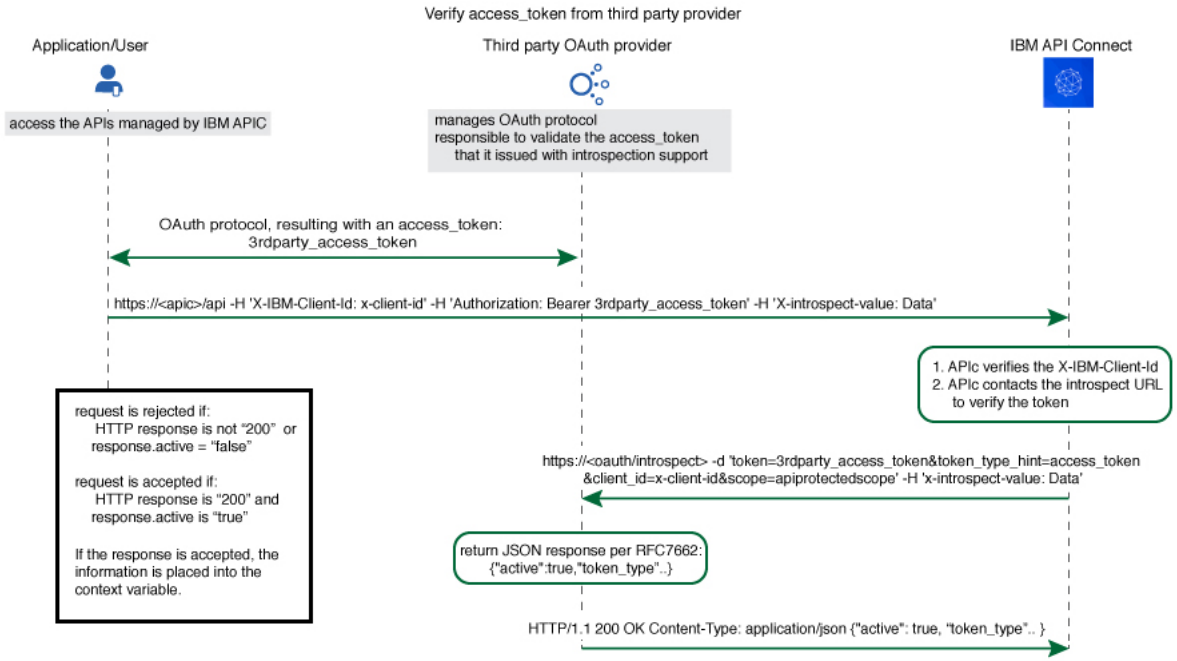
OAuth introspection for third-party OAuth providers

OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM® API Connect can use this feature along with the mentioned provider to protect access to the API.

You can use API Connect to protect an API that is secured by using the third-party OAuth access token in accordance with Introspection specification as defined in RFC 7662. In addition to the specification, the **`x-Introspect-`** header is provided to pass other content to the third party as you require.

Authentication information can be carried in the request by configuring a basic authentication request header.

The following sequence diagram depicts the overall flow of request and response. The purpose of this diagram is only to provide a general visualization of the nature of the flow; for the precise details, refer to the explanatory information after the diagram.



The Introspect URL is configured in the Third Party OAuth provider configuration. See Configuring a third-party OAuth provider.

When an API is protected by a third-party OAuth provider, API Connect will extract the bearer token and issue an HTTP POST request to the endpoint specified in the Introspect URL field.

The GET request is protected by API Connect with this feature.

You can use a header prefix to pass information to the third-party provider. The header prefix can include a regular expression and specifies the name pattern of the headers to use for sending additional information, such as **`x-introspect-*`**.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
Authorization: Bearer tGzv3JOkF0XG5Qx2TlKWIA
x-IBM-Client-Id: xxx-xxx
```

However, if you want to use a different header prefix, specify the required value in the Custom header pattern field in the third party OAuth provider configuration. For third-party OAuth provider configuration details, see Configuring a third-party OAuth provider. The Custom header pattern feature is available only with the DataPower® API Gateway, if you are using the DataPower Gateway (v5 compatible) the header prefix must be **`x-Introspect-`**.

API Connect will issue this POST request to the introspection endpoint specified in **`x-tokenIntrospect`**, as illustrated in the code sample:

```
POST /oauth/introspectURL HTTP/1.1
Host: apiconnect.ibm.com
Content-Type: application/x-www-form-urlencoded
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
token_type_hint=access_token&token=tGzv3JOkF0XG5Qx2TlKWIA
```

The third party OAuth/OIDC provider will respond with `HTTP 200` indicating the request was successful and that payload contains the token information. API Connect honors the active claim as defined in the RFC specification.

If the value of the active claim is `true`, the token is treated as valid.

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
 Cache-Control: no-store
 Pragma: no-cache
```

```
{ "active":true,
   "token_type":"bearer",
   "client_id":"xxx-xxx",
   "username":"John Smith",
   ...
}
```

If the value of the active claim is `false`, the token is treated as invalid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "active":false
}
```

A response code other than **`HTTP 200`** indicates failure to process the request.

When the OAuth token is valid and active, context variables are populated with information from the introspect JSON response. For more information, see API Connect context variables.

When contacting the introspection endpoint, API Connect uses **`client_id`**/**`appId`** and **`client_secret`**/**`appSecret`** to construct the HTTP Basic authorization header.

By default, if **`x-introspect-basic-authorization-header`** exists in the request, the value is used for the HTTP Basic authentication header when the introspection endpoint is contacted. API Connect verifies that the HTTP Basic authentication header value is Base64 encoded before it is sent, and encodes it if necessary as shown in the following example. If the header is already encoded, it is sent without modification.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-introspect-basic-authorization-header: user:password
```

API Connect issues the following:

```
POST ..
Authorization: Basic M3JkLXBhcnR5LWNsaWVudF9pZDozcmQtcGFydHlfY2xpZW50X3NlY3JldA==token_type_hint=access_token&token= ..
```

If you are using the DataPower API Gateway, you can specify your own value for the HTTP Basic authentication header by providing the required value in the Basic authentication request header name field in the third party OAuth provider configuration. The following rules determine what authentication data is passed to the introspection endpoint:

- If there is a basic authentication header in the request, the specified credentials are used. The value must be either a string in the format ***`user:password`***, or the Base64 encoded equivalent; API Connect will Base64 encode the value if necessary before sending the request to the introspection endpoint.
- If there is no basic authentication header in the request but there are values specified in the Basic authentication username and Basic authentication password fields in the third party OAuth provider configuration, those values are used.
- If there is no basic authentication header in the request, and user name and password details are not supplied in the third party OAuth provider configuration, but **`client_id`** and **`client_secret`** values are supplied in the body of the request, these are used.
- Otherwise, an error is returned.

For third-party OAuth provider configuration details, see Configuring a third-party OAuth provider.
If either, or both, of scope and **`scope validate url`** are configured, and if the response is an active token with a scope claim from the third-party OAuth Provider introspection endpoint, API Connect will further enforce the scope validation in the following order:

1. If **`scope`** is configured for the OAuth API protection, verify the third-party scope against the scope that is configured.
2. If **`scope validation url`** is configured, verify the third-party scope against the scope validation URL.
3. If the third-party OAuth provider introspection endpoint does not return a scope, the Gateway will not check the scope defined in the security definition of the API where the third-party OAuth provider resource is used. For details on configuring an OAuth security definition in an API, see Creating an OAuth security definition.

For more information, see Scope.

DataPower Gateway (v5 compatible) only By default the API Connect client ID and scope are sent to the third party OAuth provider. You can suppress this behavior in either of the following ways:

- Supply a **`suppress-parameters`** header as follows:

  `suppress-parameters: client_id`

  `suppress-parameters: scope`

  or

  `suppress-parameters: client_id scope`

  depending on which parameters you want to suppress.
- Define an API property called **`suppress-parameters`** in the API definition itself, with one of the following string values:

  `client_id`

  `scope`

  or

  `client_id scope`

  depending on which parameters you want to suppress. For information on how to define API properties, see Setting API properties.

OAuth external URL and authentication URL

You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

Including custom metadata in a token

In many scenarios, custom metadata needs to be included during the access token generation process. The metadata is either stored inside the access token or it is sent along with the access token to the client application. The client application can then send that access token, or the metadata in the payload, in a subsequent request to IBM® API Connect where the metadata is retrieved, validated, or sent to the downstream systems as required.

Examples include, but are not limited to:

- When resource owners are authenticated, metadata about the authenticated resource owner needs to be stored within the access token.
- The grant type that was used to obtain the token is another example of metadata stored within the access token.
- A confirmation code that needs to be sent to the client application along with access token is stored as metadata within the access token.

Configuring External URL or Authentication URL in API Connect to obtain metadata

Metadata can be set by using either or both of the following URLs:

- External URL - When you call the External URL, an HTTP GET request is sent and API Connect expects an HTTP 200 OK along with an optional set of the specified response headers.
- Authentication URL - When you call the Authentication URL, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.
  See: Authentication URL.

The External URL endpoint is entered in the Metadata section when you configure an OAuth Provider in the Cloud Manager or API Manager UI.

Use the following HTTP headers in the response, depending on the type of gateway you are using:

- DataPower® API Gateway:

  `X-API-OAUTH-METADATA-FOR-PAYLOAD`

  `X-API-OAUTH-METADATA-FOR-ACCESSTOKEN`

  Note: These are the default header names for the DataPower API Gateway but you can override them; see Configuring metadata for a native OAuth provider.
- DataPower Gateway (v5 compatible):

  `API-OAUTH-METADATA-FOR-PAYLOAD`

  `API-OAUTH-METADATA-FOR-ACCESSTOKEN`

The response header value from **`X-API-OAUTH-METADATA-FOR-PAYLOAD`** or **`API-OAUTH-METADATA-FOR-PAYLOAD`** is placed in the response payload and indicated as **`metadata`**.

The response header value from **`X-API-OAUTH-METADATA-FOR-ACCESSTOKEN`** or **`API-OAUTH-METADATA-FOR-ACCESSTOKEN`** is placed within the access token and indicated as **`miscinfo`**.

The two metadata response headers are case insensitive and you must escape any special characters in the string value content.

An example response payload that contains metadata along with the access token:

```
{
"token_type":  "bearer",
"access_token":  "AAEkNzhjDHYyyYy...cL0Mv6ctl37w7ZU",
"metadata":  "m:metadata-for-payload_content"
"expires_in":  3600,
"scope":  "read",
"refresh_token":  "AAEnj5SynCMybF...oEZ6JjxYax_HdNg",
}
```

This example output from the token introspection endpoint shows the contents of the access token with `"miscinfo"` containing the metadata information.

```
{
"active":  true,
"token_type":  "bearer",
"client_id":  "78c2f10f-799a-4e1f-8e0a-098634997a35",
"username":  "Fred Smith",
"sub":  "fred",
"exp":  1479850049,
"expstr":  "2016-11-22T21:27:29Z",
"iat":  1479846449,
"nbf":  1479846449,
"nbfstr":  2016-11-22T20:27:29Z",
"scope":  "read",
"miscinfo":  "m:metadata-for-accesstoken_content",
"client_name":  "MobileApp"
}
```

Input to the External URL

The following request headers are sent to the External URL.
Note: Any existing metadata values that were previously sent from the Authentication URL are also sent in the two request headers **`x-existing-metadata-for-payload`** and **`x-existing-metadata-for-access-token`**. The Metadata URL can make use of this information to create a new set of metadata values.
The two request headers that are sent to the Metadata URL are displayed in bold text.

```
X-existing-metadata-for-payload     payload-from-auth-url
X-existing-metadata-for-access-token     token-from-auth-url
X-URI-in     /org/env/miscinfo/oauth2/token (the URL that was sent to APIConnect for this particular token request)
X-METHOD-in     POST
X-POST-Body-in     client_id=client_id&grant_type=password&scope=read&username=name&password=password
X-X-Client-IP     IP_address
X-X-Global-Transaction-ID     ID_number
...
```

Note: If you are using the DataPower API Gateway rather than the DataPower Gateway (v5 compatible), the **X-POST-Body-in** header is not supported.

Retrieving Metadata in API Connect

As described in the previous example scenarios, the metadata can be retrieved from the access token in the Application API and sent to the downstream systems. Retrieval can be done in the API assembly, which is secured to accept tokens in the security definitions.

In the resource API that accepts an access token, the **miscinfo** field can be accessed in the Assembly with the **oauth.miscinfo** context variable, as in the example.

```
apim.setvariable('message.body',apim.getvariable('oauth.miscinfo'));
```

You can also use token introspection to look at the contents of the access token.

Refresh tokens and metadata

The Authentication URL (if configured for authentication) is invoked first, during authentication of the resource owner. The External URL is invoked as the last step, just before the generation of the access token. The only exception is when access tokens are generated from a refresh token. In cases where refresh tokens are used to generate new access tokens, the External URL is not invoked. The metadata from the refresh token is retained and then copied into the newly generated access token.

Identifying the source of the metadata

The metadata is prefixed with keywords to indicate whether it originated from the External URL or the Authentication URL.

- Metadata from the External URL is prefixed with **m:**
- Metadata from the Authentication URL is prefixed with **a:**

Note: When revocation is enabled, some internal details are also stored in the **miscinfo** field, in brackets within the access token as shown in the following example:

```
"miscinfo": "[tlsprofile@https://api-revoke-url:443/server/revocation-url]m:metadata-for-accesstoken_content"
```

Maximum size of the metadata

Metadata for the access token cannot exceed 512 bytes.

Metadata for the payload does not have a specific size restriction, except for when you use the Authorization code grant type. These restrictions are described in following sections.

Characters are not allowed in metadata in certain scenarios

When you use the Authorization code grant type, or when a consent form is used for implicit grant type, there is a temporary state or code where the metadata from the authentication URL is stored. API Connect internally uses two prefixes - **!ma** and **!mp** to differentiate between payload and token metadata received from the Authentication URL and store them internally in the temporary state/code. Hence these specific character sequences - **!ma** and **!mp** should not be used as the metadata itself.

Grant types and metadata

The OAuth grant types described in the following sections include **authorization code (access code)**, **implicit**, and **client credentials (application)**.

Authorization code grant type

- When metadata is included from an Authentication URL for an Authorization code grant type, as it is a three legged flow, both the content and the payload are stored within the **dp-state** and carried on to the authorization code and to the access token. Note that around 10 characters are used internally to differentiate between the metadata for payload and metadata for the access token when stored in the **dp-state**. In addition, if revocation is enabled, that will also be part of the token metadata. Hence the combined size of the token metadata, the payload metadata (including the 10 characters of internal data), and internal revocation details, cannot exceed 512 bytes in total.

  If the overall size of the metadata exceeds 512 bytes, then the access token generation succeeds, but the metadata fields contain an error message of "`metadata too large`" as shown in the example.

  ```
  "metadata":"m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-payload]"
  "miscinfo":"[r:gateway]m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-token]"
  ```

  This size restriction can be overcome when the metadata is sent from the Metadata URL and not from the Authentication URL, because the metadata is not stored in **dp-state** or in the authorization code.

  Example of the **authorization code (access code)** grant type:

  ```
   $ curl -k -d
  "grant_type=authorization_code&code=$mycode&client_id=$myid&scope=scope_introspect&redirect_uri="
  https://9.70.153.90/fei/sb/introspectp1/oauth2/token
  { "token_type":"bearer",
  "access_token":"AAEkOTh1ZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ20ZN5Tl_TqYFeIfB7BFf6HFgibEoOjWXXEA84oFsWuE4NY-
  ```

```
RRZVdnGSaXAIYJz7s5vczfk5EV-BIb_6P_1YKm3ahrfhR5kI3sPO0uADEoseIP5-O9anUpEM5yhsayXvZbJ_6VDYz-hyXSJHTNqVj-
PHBialoRkBD5qca6kO0fV2M", "metadata":"a:[Authorization Code-Test-auth-url-payload]", "expires_in":3600,
"scope":"scope_introspect", "refresh_token":"AAFAg1EVMbwicr_L0fTZ4q6HZ-
RcQygniXFC9zbSKO4wd3hcniC4KQX21X0fL2c8cnmzCZgws8xxLzNyfjQhUJNGl5C1GbIe3dwhXJdiWA0Go-
dduhVtCbG26sJRRXyYrMeRxWnJsylBETPI8HQEN4a_D7fmxKcTVRZBvq86byg95qe1ZKyERi0Lhxdd_O4Nvss" }
```

```
$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
{ "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746", "username":"anyuser",
"sub":"anyuser", "exp":1484766368, "expstr":"2017-01-18T19:06:08Z", "iat":1484762768, "nbf":1484762768,
"nbfstr":"2017-01-18T18:06:08Z", "scope":"scope_introspect", "miscinfo":"[r:gateway]a:[Authorization Code-Test-auth-
url-token]", "client_name":"oauth_app" }
```

Implicit grant type

- When implicit grant type is used, the access token and the metadata are returned in the `location` header as a fragment, as you see in the example.

```
< Location:
 https://localhost#access_token=AAEkOThlZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ2buS2KfWdq-
nYBJSi4nmPxQBtLae17tKBPRMzwP5BC386nlxpoOTE1G748ZVH6Mq_TJL3GeV3PtXTVIkWOLBJi_7tljiQfpnVfrNkovvZkhUexYmFkcDsmLSdaWxZ6Pc
IMPAC4ojT8qV1sYV-
ChTk36yqOx_NiKimZaDikDk7WTg&expires_in=3600&scope=scope_introspect&token_type=bearer&metadata=a%3A[Implicit-Test-
auth-url-payload]
```

```
$ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
"token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
             { "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746",
"username":"anyuser", "sub":"anyuser", "exp":1484768365, "expstr":"2017-01-18T19:39:25Z", "iat":1484764765,
"nbf":1484764765, "nbfstr":"2017-01-18T18:39:25Z", "scope":"scope_introspect", "miscinfo":"[r:gateway]a:[Implicit-
Test-auth-url-token]", "client_name":"oauth_app" }
```

Client credentials grant type

Authentication URL will not be invoked when using client credentials grant type, as there is no resource owner. The metadata from Authentication URL is not available for this grant type. However, content returned from Metadata URL will be included as metadata.

Behavior when retrieving metadata with both an External URL and an Authentication URL

If an External URL is configured and a connection to the external server is successful, the response headers overwrite any existing metadata obtained from the Authentication URL to become the final value. Therefore, you must carefully examine the incoming request headers and create appropriate response headers from the External URL.

If an External URL is configured, but the connection to the External URL fails, then a failure message of "`error on metadata url`" is written for metadata in both the payload and the access token.

If an External URL is configured and the connection is successful, but the remote server does not send any of the specified HTTP response headers, a blank value is written for metadata in both the payload and the access token.
Attention: An External URL overwrites existing values from Authentication URL. This includes blank values.
If no External URL is configured, the metadata that is obtained from the Authorization URL is retained as the final value.

Scope

Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.

In IBM® API Connect, scopes have no inherent meaning. Instead, scopes are defined in the OAuth Provider so that an application can request an access token that is valid for one or more of the scopes. In the secured API, scopes are listed as requirements for an access token to be considered valid. All scopes that are listed by the security definition for the API must be granted by the access token.
Note:

- An OAuth Provider must have at least one scope.
- An OAuth token will not be granted unless a default scope is configured at the provider, or one or more predefined scopes are included in the grant request.
- If no default scope is defined, and the request does not contain a scope, an invalid scope error is returned for the request.

OAuth provider

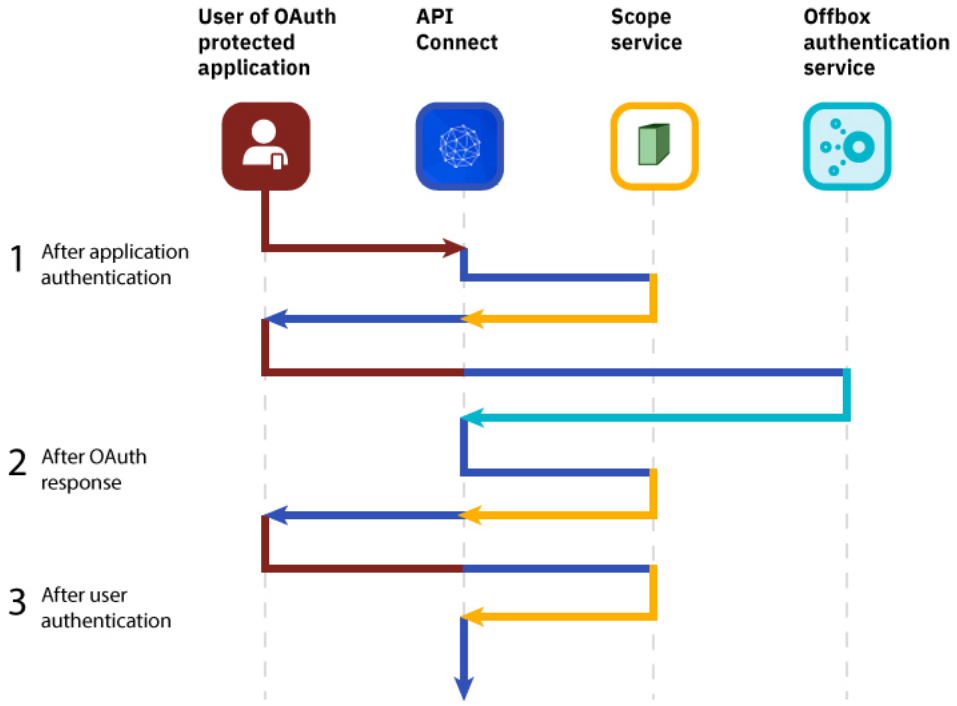
To provide more refined support for the OAuth scope handling, API Connect allows the Authentication URL user registry extension to modify the scope value.

When you define an OAuth provider, the Advanced scope check extensions provide the flexibility to check and override allowed scopes. The optional extensions are Application scope check and Owner scope check.

The scope that is eventually received by the application is determined by the interactions that are described in the three following paragraphs. Scope processing follows the sequence of items 1, 2, and 3 in the following list, offering three opportunities to override the scope value. Figure 1 provides an overview of the process.

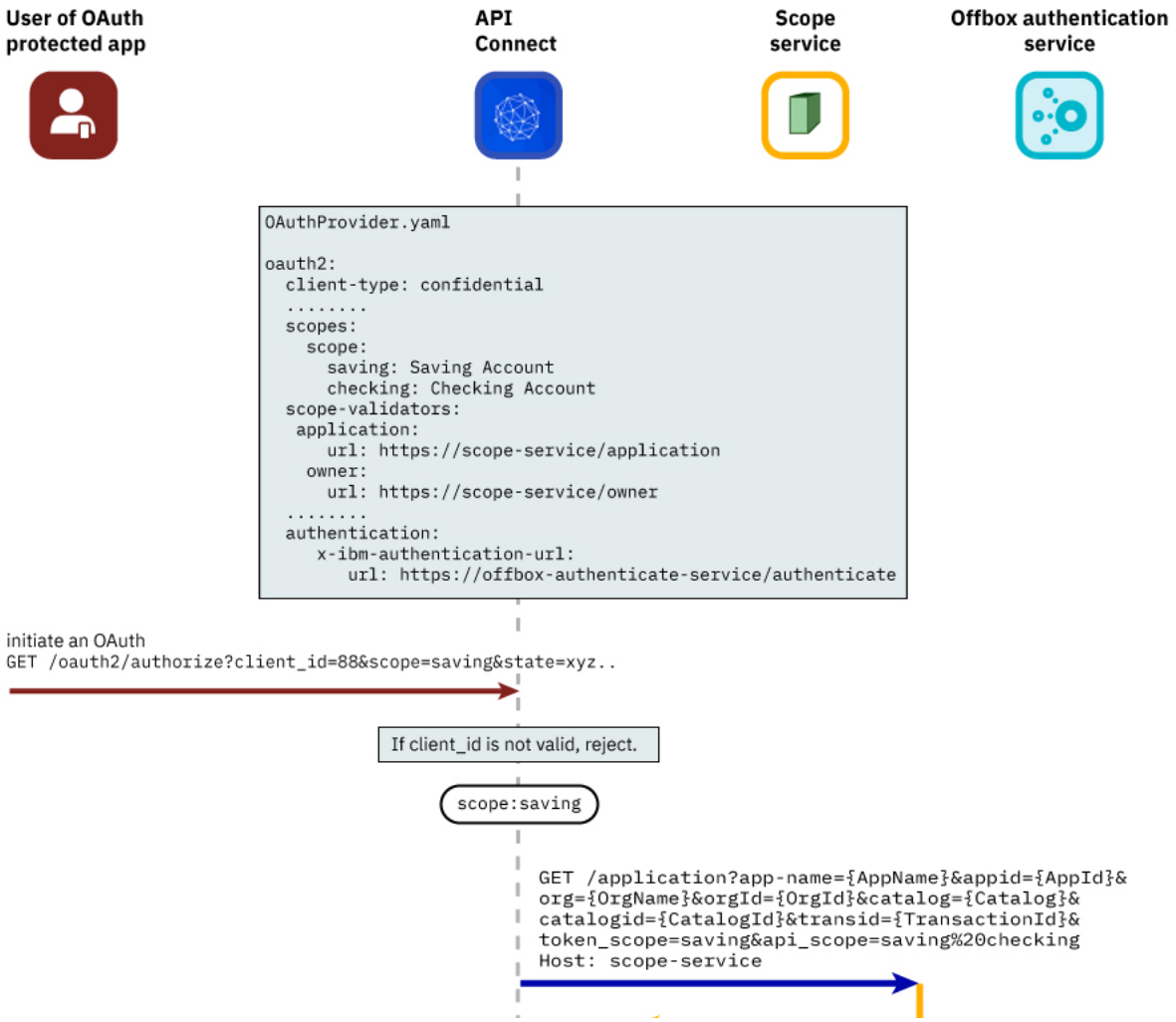
1. After the application successfully authenticates, and if OAuth Native provider > Advanced scope check > Application scope check is configured, API Connect makes a call to allow extra verification and uses the contents of **`x-selected-scope`** to override the scope value that was initially requested by the application. When Application scope check is enabled, the HTTP response header **`x-selected-scope`** must be present, or the call fails.
2. If OAuth Native provider > User Security > Authentication is configured to authenticate application users using an Authentication URL, then API Connect makes a call, as documented in Authentication URL user registry. When the response code is `HTTP 200`, and the response header **`x-selected-scope`** is present, the value that is configured in **`x-selected-scope`** is used as the new scope value, overriding both what the application already requested and what was provided in the Application scope check described in paragraph 1. In the response header, **`x-selected-scope`** is an optional element.
3. After the user successfully authenticates, and if OAuth Native provider > Advanced scope check > Owner scope check is enabled and configured with a valid URL, API Connect makes a call to allow the content of **`x-selected-scope`** to refine the scope value. When Owner scope check is enabled, the HTTP response header **`x-selected-scope`** must be present, or the call fails.

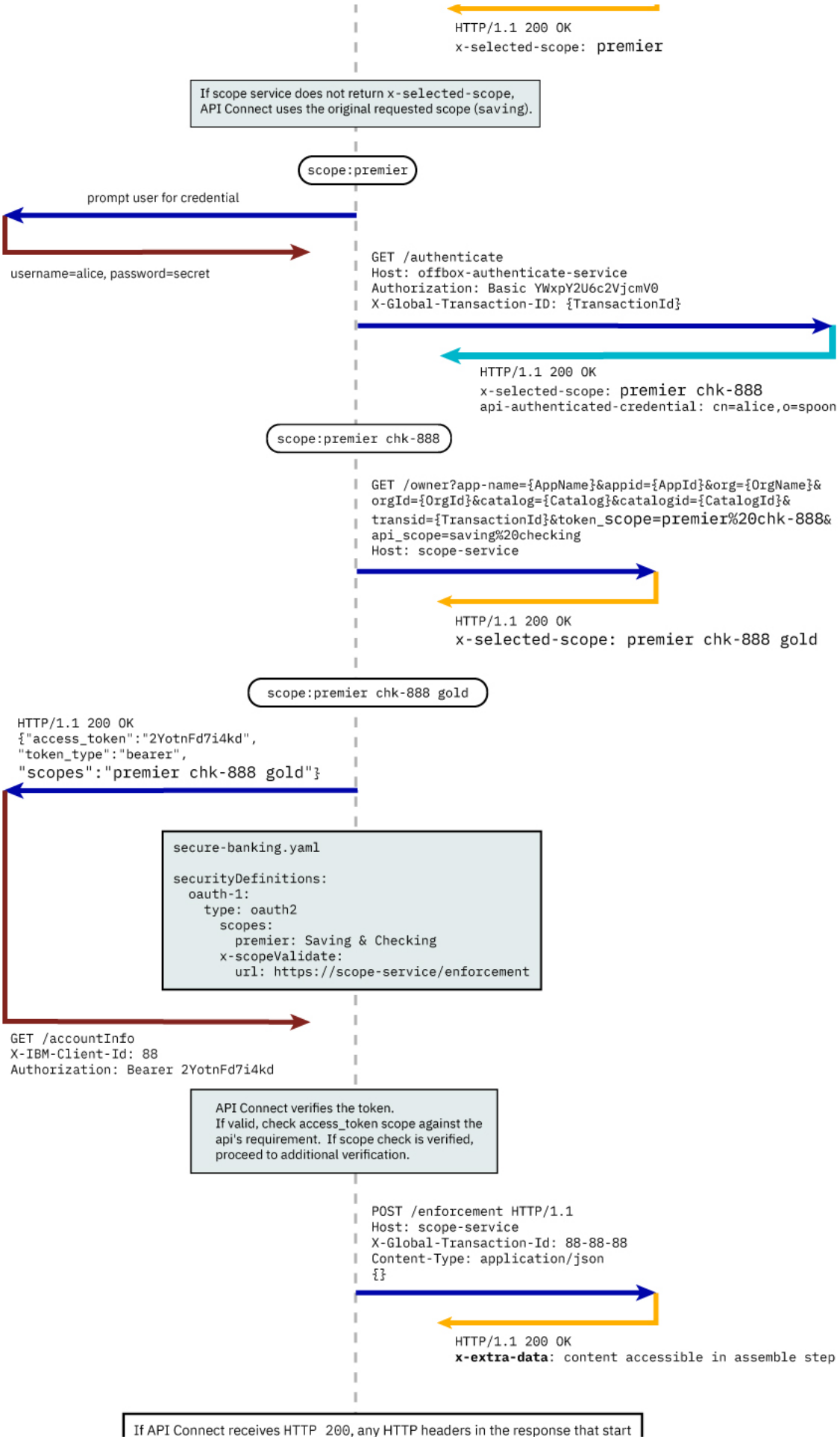
Figure 1. OAuth advanced scope overview



The final scope permission that is granted by the access token is the result of the flow described in items 1, 2, and 3. Figure 2 shows a more detailed view of the transaction flow with examples that show when **`x-selected-scope`** provides a new scope value.

Figure 2. OAuth advanced scope detail




HTTP/1.1 200 OK
x-selected-scope: premier
If scope service does not return x-selected-scope, API Connect uses the original requested scope (saving).
scope:premier
prompt user for credential
username=alice, password=secret
GET /authenticate
Host: offbox-authenticate-service
Authorization: Basic YWxpY2U6c2VjcmV0
X-Global-Transaction-ID: {TransactionId}
HTTP/1.1 200 OK
x-selected-scope: premier chk-888
api-authenticated-credential: cn=alice,o=spoon
scope:premier chk-888
GET /owner?app-name={AppName}&appid={AppId}&org={OrgName}&
orgId={OrgId}&catalog={Catalog}&catalogid={CatalogId}&
transid={TransactionId}&token_scope=premier%20chk-888&
api_scope=saving%20checking
Host: scope-service
HTTP/1.1 200 OK
x-selected-scope: premier chk-888 gold
scope:premier chk-888 gold
HTTP/1.1 200 OK
{"access_token":"2YotnFd7i4kd",
"token_type":"bearer",
"scopes":"premier chk-888 gold"}
secure-banking.yaml
securityDefinitions:
oauth-1:
type: oauth2
scopes:
premier: Saving & Checking
x-scopeValidate:
url: https://scope-service/enforcement
GET /accountInfo
X-IBM-Client-Id: 88
Authorization: Bearer 2YotnFd7i4kd
API Connect verifies the token.
If valid, check access_token scope against the api's requirement. If scope check is verified, proceed to additional verification.
POST /enforcement HTTP/1.1
Host: scope-service
X-Global-Transaction-Id: 88-88-88
Content-Type: application/json
{}
HTTP/1.1 200 OK
x-extra-data: content accessible in assemble step
If API Connect receives HTTP 200, any HTTP headers in the response that start

with "x-" will be accessible at oauth.advanced-consent.<value>
For the above example, x-extra-data is accessible at
oauth.advanced-consent.x-extra-data.
If other than HTTP 200 is returned, it has the same effect as if the access token does not contain neccesary permissions to access the resource.

Consumer API enforcement

Standard scope validation
To access the API /getaccount the application must send a **GET** request with an access token that contains the scope, or scopes, defined in the OAuth provider. If the request doesn't contain a scope, you must specify a default scope to use. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

```
GET /getaccount
HTTP/1.1
Host: server.example.com
X-IBM-Client-Id: 8888-8888-8888
Authorization: Bearer AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ
```

The following application OpenAPI file secure-banking.yaml defines the scope, or scopes, that must exist in the token to be granted access to the API /getaccount.

```
secure-banking.yaml

securityDefinitions:
   scope-only:
    type: oauth2
    description: ''
    flow: implicit
    authorizationUrl: ''
    scopes:
      checking: 'Checking Account'
      saving: 'Saving Account'
      mutual: 'Mutual Fund Account'
security:
  - scope-only:
      - checking
  - scope-only:
      - saving
      - mutual
```

In the examples, the access token **AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ** is able to access the API because it contains one, or a combination of **scope-only** that is defined in secure-banking.yaml such as:

- **checking**
- **saving mutual**
- **checking saving mutual**

Advanced Scope Check
Administrators can enable an additional scope check by configuring the consumer API property Advanced Scope Check URL that becomes **x-scopeValidate** as shown in the following OpenAPI file example.

```
securityDefinitions:
  advanced-scope-only:
    type: oauth2
    description: ''
    flow: implicit
    authorizationUrl: ''
    scopes:
      checking: 'Checking Account'
      saving: 'Saving Account'
      mutual: 'Mutual Fund Account'
    x-scopeValidate:
      url: 'https://advanced-scope-check.bk.com/validate-scope'
      tls-profile: 'ssl-client'
```

After API Connect successfully verifies the access token against any scope requirement, API Connect will make an **HTTP POST** request to the **x-scopeValidate** endpoint similar to the following example. Response code HTTP 200 from the endpoint indicates a success. Any other response code, or a connection error, is treated as a permission error for the token.

```
     POST /validate-scope?app-name=..&appid=..&org=..&orgid=..&catalog=..&catalogid=..&transid=..
HTTP/1.1
     Host: advanced-scope-check.bk.com
     Content-Type: application/json

   {"context-root" : checking,
    "resource" : accountinfo,
    "method" : GET,
    "api-scope-required" : [jointaccount],
    "access_token" : {"client_id" : "2cd71759-1003-4a1e-becb-0474d73455f3",
                      "not_after" : 174364070,
                      "not_after_text" : "2017-07-11T02:27:50Z",
                      "not_before" : 174360470,
                      "not_before_text" : "2017-07-11T01:27:50Z",
                      "grant_type" : "code",
                      "consented_on" : 1499736470,
                      "consented_on_text" : "2059-07-11T01:27:50Z",
                      "resource_owner" : "cn=spoon,email=spoon@poon.com",
                      "scope" : "jointaccount mutual",
                      "miscinfo" : "[r:gateway]"
```

```
                    }
}
```

An example of successful response follows.

```
    HTTP/1.1 200 OK
    Cache-Control: no-store
    Pragma: no-cache
    X-Custom-For-Assemble-Process: audit
```

Any HTTP response header that begins with "**x-**" is kept as the context variable **oauth.advanced-consent**. Based on the example successful response, **X-Custom-For-Assemble-Process: audit** becomes `oauth.advanced-consent.x-custom-for-assemble-process`, and can be accessed in the assemble step.

Additional validation options
You can optionally use additional fields for validation:

Request Header
: Defines the regular expression to match against **request** headers. Matching headers are included in the request to the advanced scope validation endpoint.

Response Context Variable
: Defines the regular expression to match against **response** headers. Matching headers are saved as context variables in the format **oauth.advanced-consent.***.

Related information

- IETF RFC 6749

Tokens

Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.

This section contains information regarding token management, including refreshing and revoking tokens, redirecting, and managing tokens with the DataPower Gateway.

- **Refresh tokens**
  If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.
- **Token revocation**
  In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.
- **Authenticating and authorizing through a redirect URL**
  You can use a service that is hosted externally from IBM API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.
- **Token management with the DataPower Gateway (v5 compatible)**
  API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.
- **Token management with the DataPower API Gateway**
  API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Refresh tokens

If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.

When you are using OAuth authentication, API requests must include a valid access token, by using the Authorization HTTP header. Access tokens that are issued by the IBM® API Connect Token Endpoint are valid for 3600 seconds (1 hour) by default, as indicated by the expires_in property that is returned on the token request. The following code block shows an example API request with an Authorization header:

```
GET /bankingApi/accountSummary?client_id=32427ce5-bb7c-48a7-9de3-4bb629091103
HTTP/1.1
Accept: application/json
Host: api.ibm.com
Authorization: Bearer AAEFYy1hbGxlhdS5nVX4x6iTL2sb3ymBivQb...
```

After an access token expires, if the option is enabled in the OAuth provider configuration Tokens > Refresh tokens screen, the application uses refresh tokens. Each refresh token is valid for approximately 31 days after it is issued (or for the Time to Live time period specified) and can be used only once to request a new access token. Along with the new access token, a new refresh token is also returned. For details on how to enable refresh tokens, see Configuring a native OAuth provider.

If the access token is expired and the application does not have a refresh token, it must restart the OAuth exchange by using the choice of Grant Type(s) allowed by the OAuth provider.

Token revocation

In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.

Token revocation using an external third party service

This topic describes token revocation using a third party, external service. This option is configured in the Native OAuth provider, using the Token Management > Type = Third party screen. The revocation URL is an endpoint that links to an external service which contains information about access or refresh tokens. API Connect is involved in the initial creation and validation of tokens. When an OAuth revocation URL is present, API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing.

Examples

A number of revocation examples follow. The first shows a sample fetch request and the response from a remote revocation URL.

GET request and response

In this example, an API Gateway server issues a GET request to the Revocation URL and receives a result. It shows that different resource owners (Laura and Emily) can revoke all tokens when using the same application (client ID).

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
GET <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Accept: application/xml; text/xml
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>

<!--
Access Tokens and/or Refresh Tokens that are revoked can be individually listed.
To keep this list small, please only include access tokens and refresh tokens that are valid.
For access tokens, any token older than 20 minutes is no longer valid.
For refresh tokens, any token older than 44700 minutes is no longer valid.
-->

<token type="access">AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1...</token>
<token type="refresh">fZaRlVbnPSc1UGTjCRdq4mPbOosD2+aZIKbJ6bTeW...</token>

<!--
If a resource owner has revoked all tokens issued to a given application, please
list them as shown here.
-->

<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">laura</resource-owner>
<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">emily</resource-owner>

</oauth-revocation>
```

POST request and response

The next example shows a post request and response.

Request:

```
POST <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Content-Type: application/xml

<?xmlversion="1.0" encoding="UTF-8"?>
<token>
   <token_type>bearer</token_type>
   <access_token>AAENYy1hbGwtcmVmcmVzaOfNeQKX8ZeojsBY9v0FI7/OerQvzKHq...</access_token>
   <expires_in>3600</expires_in>
   <scope>3600</scope>
   <resource-owner>alice</resource-owner>
   <client_id>83d9cdcd-ba72-4d00-abae-005da8da5fb1</client_id>
</token>
```

Response:

```
HTTP/1.1 200 OK
```

Provide token information on revocation request

In this example, the application calls an API and passes a bearer token. In response, the Gateway fetches the revocation URL and provides information on the token being verified.

Gateway:

```
GET  <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Alternatively, the same process occurs when using a refresh token to issue a new access token. The application sends a refresh request to the token service. The Gateway then fetches the revocation URL, providing information on the refresh token being verified.

Gateway:

```
GET  <revocationURL>GET  HTTP/1.1
refresh-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Note: If you are using a third party OAuth provider then, for API Calls with bearer tokens, when Introspect URL is enabled on the API, the revocation URL is not applicable. Instead, the third party endpoint must validate the token and also check for revocation before returning a `200 OK` response to the Gateway.

Revoking tokens issued to Alice up to and including a specific date

On May 1st, Alice loses her phone and needs to reset her password. As a result, the token provider wants to revoke every token issued before Alice lost her phone. In this example, the Gateway sends a `GET` request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
  <resource-owner before="2015-05-01T09:30:10Z">alice</resource-owner>
<oauth-revocation>
```

Revoking all tokens issued up to and including a specific date

Under certain catastrophic conditions, you may need to revoke all tokens issued up to and including a specific date, for example, May 1st. In this example, the Gateway sends a `GET` request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
  <everytoken before="2015-05-01T09:30:10Z" />
<oauth-revocation>
```

Putting it all together

The following shows the examples contained in this topic executed in a single action:

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Cache-Control: public, max-age=120
Date: Fri, 08 May 2015 21:49:03 GMT


<?xml version-"1.0" encoding="UTF-8"?>
<oauth-revocation>
  <resource-owner before="2015-04-08T09:30:10Z">mary</resource-owner>
  <resource-owner before="2015-04-12T09:30:10Z">john</resource-owner>
  <resource-owner before="2015-04-13T09:30:10Z">kevin</resource-owner>
  <resource-owner before="2015-04-01T09:30:10Z">alice</resource-owner>
<oauth-revocation>
```

Notes:

- In the previous example, there are no entries older than one month in the response (the maximum life of a refresh token).
- Each response is cached for up to two minutes according to response's directive.
- The `before` attribute uses the `xs:dateTime` syntax.

Authenticating and authorizing through a redirect URL

You can use a service that is hosted externally from IBM® API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.

Before you begin

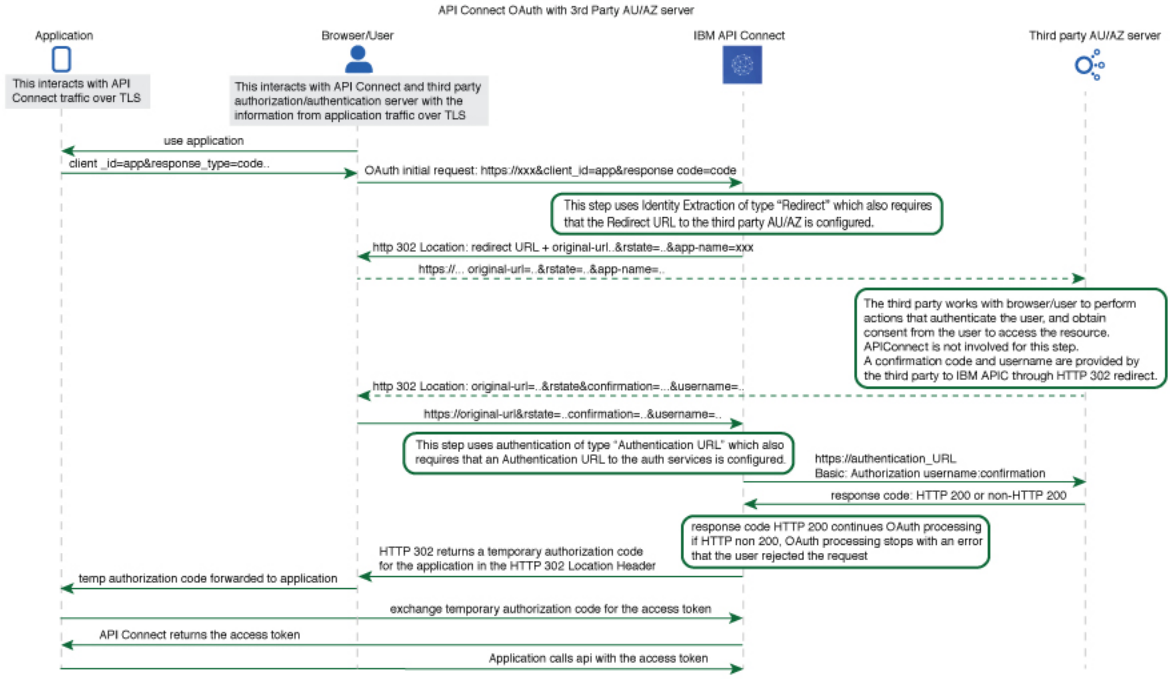
To complete this task, you will need to either create or have created an OAuth security definition that uses Implicit grant type or Access Code (Authorization Code) grant type. For more information, see Creating an OAuth security definition.

About this task

If you use methods for authentication that are not supported by API Connect, you can redirect users to a suitable URL at which they can authenticate. The user is then returned to the OAuth process after authentication and authorization have been confirmed.

The following illustration indicates the transaction flow for third party authentication.

Figure 1. Third party authentication (AU) and authorization (AZ) transaction flow



1. The application initiates a request to access an API protected with a third-party entity. API Connect redirects the application with an **`HTTP 302`** redirect based on **`identity extraction -> redirect -> redirect-url`**, for user authentication (and optional authorization).
2. The application communicates directly with the third-party entity to gather user identity. API Connect is not involved in this communication. After the third-party entity finishes processing authentication (and optional authorization), it returns an **`HTTP 302`** redirect that uses the **`original-url`** from the request, with the username and confirmation code appended.
3. API Connect receives the request that includes the username and confirmation code, and communicates with the authentication URL, based on **`authentication -> x-ibm-authentication-url`**, to confirm user identity before the request is completed.
4. An **`HTTP 200`** response from the third-party entity allows API Connect to continue the OAuth 2.0 request process as if the owner is authenticated. The request is then processed according to the **`grants`** type.
   - **`- accessCode`** returns a temporary code to the application.
   - **`- implicit`** returns the access token to the application.

   For any response other than HTTP 200, the request fails with a statement added to the error log.

Procedure

To create an external form, and to indicate the URL to which API Connect will redirect users, complete the following instructions:

1. Create your service for authentication and authorization. You will use the URL of the landing page of this service as your redirect URL.
   a. To include elements in your form that are provided by API Connect, use the following query parameters from the URL that your user is redirected to.
   Note: With the exception of **`original-url`**, none of these parameters are included in the URL automatically; you must add them as required.

   app-name
   : The name of the application requesting access, as provided through the Developer Portal.

   appid
   : The id of the application requesting access.

   catalog
   : The name of the catalog where the product is being used by the application.

   catalogid
   : The id of the catalog where the product is being used by the application.

   catalogtitle
   : User-friendly display name for the catalog.

   org
   : The name of the consumer organization that hosts the application.

   orgid
   : The id of the consumer organization that hosts the application.

   orgtitle

User-friendly display name for the organization.

original-url
: The original URL that the user was directed to by the application, including query parameters from the original URL that are necessary for standard OAuth 2.0 requests. You can include these parameters in your service to provide information to the user. Additionally the state_nonce is appended. The state_nonce is a hash code generated by API Connect for verification purposes. The URL is URL-encoded and should be decoded before further use, the **`state_nonce`** should remain unchanged.

provider
: The name of the API provider organization.

providerid
: The id of the API provider organization.

providertitle
: User-friendly display name for the provider organization.

requested-scope
: [optional] If Application Scope check is enabled and replaces the **`scope`** from the initial application request, this field holds the **`scope`** value from the initial application request, and the new replacement scope value is put into **`original-url`**.

transid
: transaction id used in the GW for the transaction which trigger this call

The URL to which the user is sent to when they are redirected to your page has the following form:

```
Redirect_URL?original-url=Original_URL&state_nonce=R_State&app-name=Application_Name
```

where all variables are as described previously. The Redirect URL does not have a size limit enforced by API Connect.

b. Create the stages of authentication, authorization, and any intermediate stages that you require to take place before you allow access to the application. Upon completion of these stages, redirect the user to the *Original_URL* and append a user name, their confirmation code, and the application name to be evaluated for access grant or denial by API Connect. The confirmation code does not have a size limit enforced by API Connect.
Original URL requires the following form:

```
Original_URL&username=User_Name&confirmation=Confirmation_Code
```

where all variables are as described previously.

For example:

```
https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
5709432af66e&state_nonce=HoIbRG+6bZtqlB7LDkq4gjlD3SHKglCbnYdHs/bMz2Y=&username=spoon&confirmation=12345678
```

c. To send your own error responses after the authentication and authorization service, redirect the user to the *Original_URL* and append an error code. You can also append a user name. Use the following form:

```
Original_URL&username=User_Name&error=Error_Response
```

where *Error_Response* is the message you wish to send and all other variables are described as previously.

For example:

```
https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
5709432af66e&state_nonce=HoIbRG+6bZtqlB7LDkq4gjlD3SHKglCbnYdHs/bMz2Y=&username=User&error=access_denied
```

2. Create a service to validate the confirmation code and user name. API Connect makes a GET call to your authentication URL after the user is redirected back to the authorization URL. When the call is made, it includes in its authorization header the user name and confirmation code you supplied previously. Confirm that these are correct and respond with an HTTP success code such as `200 OK` if you want to allow access, or non-`200 HTTP` response code, such as `401 Unauthorized` to deny access.
3. In your OAuth provider configuration, supply the redirect URL that is used in Step 1 and the authentication URL that is used in Step 2.
For more information on configuring an OAuth Provider, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Token management with the DataPower Gateway (v5 compatible)

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

In order to manage tokens with the DataPower® Gateway (v5 compatible), you must set the Token Management Type to Native in your Native OAuth provider configuration. See Configuring token management and revocation for a native OAuth provider when using Cloud Manager or Configuring token management and revocation for a native OAuth provider when using API Manager.

Also, the DataPower quota enforcement server must be enabled on the DataPower Gateway to use the distributed cache to manage tokens. See Quota enforcement.

Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

When distributed cache support is enabled, replay protection is provided across the gateway cluster through the quota enforcement server. This support ensures that the same token cannot be reused across the members of the quota enforcement peer group.

Important:
If you have a cluster of DataPower Gateway servers, the OAuth data synchronization behavior across the servers depends on whether or not you enable revocation:

- If you enable revocation, API Connect uses the DataPower quota enforcement server, and OAuth data is synchronized across the servers. If an access token is obtained from one server, the OAuth data synchronization ensures that the same authorization code cannot be used to obtain an access code from another server. You must ensure that the DataPower quota enforcement server is configured.

- If you disable revocation, API Connect does **not** use the DataPower quota enforcement server and OAuth data does not synchronize across the cluster of DataPower Gateway servers. Therefore, to prevent the same authorization code being used to obtain an access code from more than one server you must configure DataPower to synchronize OAuth data across the servers, by using the DataPower Gateway console.

To configure DataPower to synchronize OAuth data, ensure that the DataPower quota enforcement server is configured. For more information, see Configuring the quota enforcement server.

Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
       Host: apic.ibm.com
       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
       Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

[

{

"clientId": "5287fe53-8747-438a-8262-681ec75b79c5", => same client id as request

"owner": "cn=spoon,o=ibm",

"clientName": "PetStore Application",

"scope": "listpet",

"issuedAt": 1503327054,

"consentedOn": 1503327054,

"expiredAt": 1503330654,

"refreshTokenIssued": false,

"appId": "d2031f0f27339315333734ab9",

"org": "PetStoreOrg",

"orgTitle": "Katie Pet Grooming Inc",

"orgId": "5887803de4b06e6998c4b2c7",

"provider": "SuperStore",

"providerTitle": "Simon SuperStore",

"providerId": "5887803de4b06e6998c4b2c7",

"catalog": "publicapi",

"catalogTitle": "For public",

"catalogId": "5887803de4b06e6998c4b2d3"

} => delete second item

]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8

HTTP/1.1
```

```
Host: apic.ibm.com

x-ibm-client-id: a8746323-9825-a842-8736-abd8202356ac8 => same client id as delete request

x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7

Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

    { "status": "success" }
```

Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 IETF RFC 7009. An HTTP POST operation that an application can send to this API to revoke either an **access_token**, or **refresh_token** with **token_type_hint** as shown in the following examples:

Revoke **access_token**

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded

     token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn94lSWX14xuR....
```

- Response:

```
HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: private, no-store, no-cache, must-revalidate
     Pragma: no-cache

    { "status": "success" }
```

Revoke **refresh_token**

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded

     token_type_hint=refresh_token&token=........
```

- Response:

```
HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: private, no-store, no-cache, must-revalidate
     Pragma: no-cache

    { "status": "success" }
```

Token management with the DataPower API Gateway

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Note: The steps described in this topic are required if you are using the DataPower® OVA or appliance version of the gateway. For the API Connect supplied Kubernetes deployment, the API Security Token Manager is preconfigured provided that, in the gateway service custom resource file, v5 compatibility mode is disabled and the token management service is enabled:

```
      .
      .
      .
spec:
      .
      .
      .
  apicGatewayServiceV5CompatibilityMode: false
```

```
  tokenManagementService:
    enabled: true
      .
      .
      .
```

For more information, see Installing the DataPower Gateway subsystem.
Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

In order to manage tokens with the DataPower API Gateway, you must set the Token Management Type to Native in your Native OAuth provider configuration. See Configuring token management and revocation for a native OAuth provider when using Cloud Manager or Configuring token management and revocation for a native OAuth provider when using API Manager.

Note: This setting is also required when you use a third-party OAuth provider and **`Introspect cache type`** is set to **`protocol`** or **`Time to live`**.
For token management with the DataPower API Gateway, you must configure the API Security Token Manager on the gateway. Complete the following steps:

1. Log in to the DataPower administration console, selecting apiconnect for the domain, and WebGUI for the Graphical Interface.
2. In the search box, enter API Security Token Manager, then click on the API Security Token Manager link that displays in the search results.
3. For the Administrative state, select enabled.
4. Click the + icon alongside the Gateway Peering field to create a new gateway peering object.
5. Provide a Name and a Local address.
6. In the Local port and Monitor port fields, provide values that aren't already in use by services for the Local address.
7. Ensure that Peer group mode is selected.
8. Alongside the Peers field, use the add button and accompanying field to add at least two peers, to ensure that quorum is achieved.
9. If the Enable SSL option is selected, select a value for the Identification credentials.
10. For Persistence location, select a setting other than memory.
11. When done, click Apply.
12. Repeat steps 1 to 11 for each DataPower API Gateway in the peer group.
13. When done, click Apply, then click Save Configuration to save your changes.

For more information, see Defining the API security token manager and Creating a gateway peering instance.

Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user **`cn=spoon,o=ibm`** with username **`spoon`** and password **`spoon`** using a registered administration application of **`5287fe53-8747-438a-8262-681ec75b79c5`**.

- Request:

```
GET /oauth2/issued
HTTP/1.1
       Host: apic.ibm.com
       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
       Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

[

{

"clientId": "5287fe53-8747-438a-8262-681ec75b79c5", => same client id as request

"owner": "cn=spoon,o=ibm",

"clientName": "PetStore Application",

"scope": "listpet",

"issuedAt": 1503327054,

"consentedOn": 1503327054,

"expiredAt": 1503330654,

"refreshTokenIssued": false,

"appId": "d2031f0f27339315333734ab9",
```

```
"org": "PetStoreOrg",

"orgTitle": "Katie Pet Grooming Inc",

"orgId": "5887803de4b06e6998c4b2c7",

"provider": "SuperStore",

"providerTitle": "Simon SuperStore",

"providerId": "5887803de4b06e6998c4b2c7",

"catalog": "publicapi",

"catalogTitle": "For public",

"catalogId": "5887803de4b06e6998c4b2d3"

} => delete second item

]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8

HTTP/1.1

Host: apic.ibm.com

x-ibm-client-id: a8746323-9825-a842-8736-abd8202356ac8 => same client id as delete request

x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7

Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

    { "status": "success" }
```

Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 IETF RFC 7009. An HTTP POST operation that an application can send to this API to revoke either an **access_token**, or **refresh_token** with **token_type_hint** as shown in the following examples:

Revoke **access_token**

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded

     token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn94lSWX14xuR....
```

- Response:

```
HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: private, no-store, no-cache, must-revalidate
     Pragma: no-cache

    { "status": "success" }
```

Revoke **refresh_token**

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded
```

```
      token_type_hint=refresh_token&token=........
```

- Response:

```
HTTP/1.1 200 OK
      Content-Type: application/json;charset=UTF-8
      Cache-Control: private, no-store, no-cache, must-revalidate
      Pragma: no-cache

     { "status": "success" }
```

Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes **`cn=spoon,o=eatery`**. The credential is kept in the OAuth **access_token** to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the **`access_token`**.
- Overriding the **`scope`** that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the **`scope`** value that the application receives. For example, you can provide a specific resource owner an account number within the **`scope`** header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use **`cn=spoon,o=eatery`** as the user identity.

```
HTTP/1.1 200 OK
Server: example.org
X-API-Authenticated-Credential: cn=spoon,o=eatery
```

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see User Security policy.

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see Example - using multiple OAuth policies in an OAuth provider assembly.

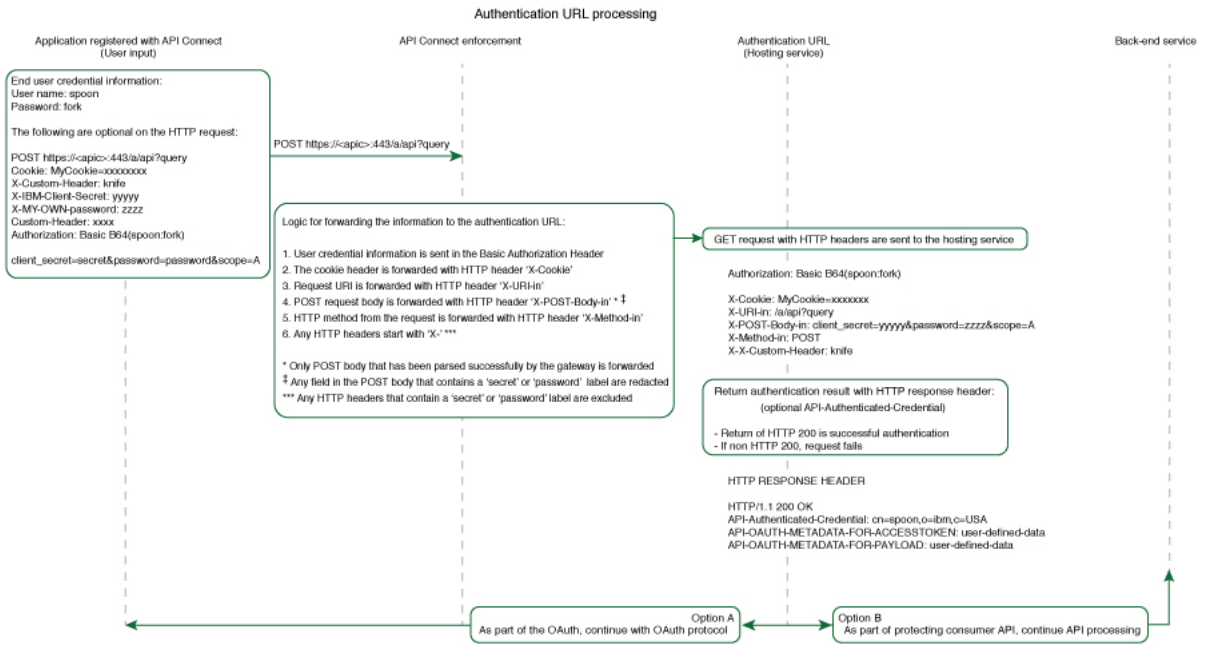
API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see OAuth external URL and authentication URL. The two metadata response headers are:

```
API-OAUTH-METADATA-FOR-ACCESSTOKEN
API-OAUTH-METADATA-FOR-PAYLOAD
```

When an Authentication URL is invoked, an HTTP response header is available to override the requested **`scope`** from the application. For more information, see Scope. The response header is:

```
x-selected-scope
```



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

Custom forms for user security

You can create custom forms for the authorization and identity extraction phase of OAuth.

About this task

The Native OAuth provider configuration provides the capability for requiring additional authentication and authorization steps for user security. Custom HTML forms may be created to extract the identity of the user and to authorize the user. This capability applies to Implicit, Resource owner password, and Access code grant types.

- **Creating a custom HTML login form for user security**
  Custom HTML forms can be created for user security during the identity extraction stage in OAuth.
- **Creating a custom HTML authorization form for user security**
  Custom HTML forms can be created for user security during the authorization stage in OAuth.

Creating a custom HTML login form for user security

Custom HTML forms can be created for user security during the identity extraction stage in OAuth.

Before you begin

The Native OAuth provider configuration includes Identity Extraction when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to extract the user credential and one of the choices is Custom HTML Form. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager. For more information, see Configuring a native OAuth provider. This topic describes how to create the Custom HTML form for identity extraction.

About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form for signing in to the service provided by the API. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

Procedure

To create a custom sign-in form for your Native OAuth provider, complete the following steps:

1. Create a well formed XHTML document that will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as `POST`, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.
   For example:

   ```
   <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
   ```

3. Create a text input field that is named `username` and create a password input field named `password`.
4. Add the line `<EI-INJECT-HIDDEN-INPUT-FIELDS>`. This third element is a placeholder that API Connect replaces with input fields to complement the user-submitted data.
5. Create a button to initiate the sign-in process.
   For example:

   ```
   <button
   id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.apionprem.doc_oauth_custom_login_form_login_button" type="submit" name="login" value="true">Log in</button>
   ```

6. Optional: Add text that is displayed the first time that the user visits the sign-in page. Use the tag `<EI-LOGINFIRSTTIME>` for the text that you want to display.
7. Optional: Add text that appears when the user is returned to the sign-in page if they fail to authenticate. Use the tag `<EI-LOGINFAILED>` for the text that you want to display.
8. Optional: Have an error message displayed when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<EI-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
9. Optional: You can add elements that are loaded from external sources, such as images or JavaScript.
10. Insert spacing and other features as you require. Completing Steps 1 through to 8 results in a form similar to the following example:

    ```
    <html lang="en" xml:lang="en">
      <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/></head>
      <body>
        <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
          <h1>Please sign in</h1>
          <p>Username </p>
          <p ><input type="text" name="username" required="required" /> </p>
          <p>Password </p>
          <p ><input type="password" name="password" required="required" /> </p>
          <EI-INJECT-HIDDEN-INPUT-FIELDS/>
          <p > <button
    ```

```
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.apionprem.doc_oauth_custom_login_form_login_bu
tton" type="submit" name="login" value="true">Log in</button> </p>

      <EI-LOGINFIRSTTIME>
        <p>If you have forgotten your user name or password, contact your system administrator.</p>
      </EI-LOGINFIRSTTIME>

      <EI-LOGINFAILED>
        <p >At least one of your entries does not match our records.
          If you have forgotten your user name or password, contact your system administrator.</p>
      </EI-LOGINFAILED>

      <EI-INTERNAL-CUSTOM-FORM-ERROR/>

    </form>
  </body>
</html>
```

11. Make your form available at a URL of your choice.
12. If you have not already done so, configure your Native OAuth provider to use a Custom HTML form for identity extraction and provide the URL at which your form is available. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Related tasks

- Creating a custom HTML authorization form for user security

Related information

- Configuring API security

Creating a custom HTML authorization form for user security

Custom HTML forms can be created for user security during the authorization stage in OAuth.

Before you begin

The Native OAuth provider configuration includes user authorization when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to authorize application users and one of the choices is Custom HTML Form. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager. This topic describes how to create the Custom HTML form for authorization.

About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form through which they grant permission to an application to access their data through the API on their behalf. You can present a custom form or a default form. Your custom form must fulfill certain requirements.
Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

Procedure

To create a custom authorization form for your Native OAuth provider, complete the following steps:

1. Create a well-formed XHTML document. This will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as `POST`, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.
   For example:

   ```
   <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
   ```

3. Add the line `<AZ-INJECT-HIDDEN-INPUT-FIELDS/>`. This line is a placeholder that API Connect will replace with input fields necessary for the completion of the OAuth process.
4. Create two buttons with the following code so that the user can grant or deny permission. Edit the text to suit your preferences.

   ```
   <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
   <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
   ```

5. Optional: Display an error message when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<AZ-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
6. Optional: You can add to the form HTML elements that will load features from external sources, such as images or JavaScript.
   For example, `<script src="http://www.example.com/example.js" />`
7. Insert spacing and additional elements as you require. Completing Steps 1 through to 6 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
  <head><title>Request for permission</title></head>
  <body class="customconsent">
    <div>
      <div>
        <form method="post" enctype="application/x-www-form-urlencoded" action="authorize">
          <AZ-INJECT-HIDDEN-INPUT-FIELDS/>
```

```
          <p>Greeting..</p><DISPLAY-RESOURCE-OWNER/>
          <p>This app </p><OAUTH-APPLICATION-NAME/><p> would like to access your data.</p>
          <div>
            <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
            <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
          </div>
        </form>
      </div>
      <AZ-INTERNAL-CUSTOM-FORM-ERROR/>
    </div>
  </body>
</html>
```

8. Make your form available at a URL of your choice.
9. If you have not already done so, configure your Native OAuth provider to use a Custom HTML Form for authorization for User Security. Provide the URL as the endpoint at which your form is available. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Related tasks

- Creating a custom HTML login form for user security

Related information

- Configuring API security

Securing an API with a JSON Web Token

There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM® API Connect.

About this task

JSON Web Token (JWT) is an OAuth 2.0 compliant method of authentication that can be useful to secure your API in API Connect.

You can secure your API with a JSON Web Token by using either of the following methods:

- Generate a token through the **jwt-generate** command, and then augment the response payload with your generated token replacing the `id` token.
- Use a token that was generated outside of API Connect and include it into the response payload, by using the metadata URL.

Procedure

Create a **jwt-generate** policy in the assembly.

a. In API Manager, open the Assembly tab.
b. Add a jwt-generate policy to the assembly for the API.

Troubleshooting OAuth

You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

You can use the following links to go to the topics:

- OAuth 2.0 support in the developer portal test tool
- OAuth requires quorum in a multi-node cluster

OAuth 2.0 support in the developer portal test tool

OAuth 2.0 support is exposed as an API through the provider OpenAPI definition. You can use the test tool that is included with API Connect to test the OAuth 2.0 configuration. For more information, see Testing an API using the Developer Portal test tool.

OAuth requires quorum in a multi-node cluster

In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

Configuring the cloud settings

Before you can add a provider organization to your cloud, you must specify your cloud settings to configure an email server for notifications as well as user registry options and catalog defaults.

The cloud settings let you define basic configuration for your deployment. To view the current settings, log in to Cloud Manager and click in the navigation pane.

Tip: When you are familiar with the cloud settings, you can modify them using the Toolkit command-line interface instead of the user interface.
Visit the following pages to configure settings:

Overview
: Provide a title and description for your cloud administrator organization. For more information, see Change the name of your cloud.

Onboarding
: Specify time-out settings for onboarding invitations, password resets, access tokens, and refresh tokens. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
For more information on setting timeouts, see the following topics:

  - Configuring invitation timeouts
  - Configuring the password-reset notification and timeout
  - Configuring timeouts for access tokens and refresh tokens

User Registries
: Configure one or more user registries to authenticate users of Cloud Manager or API Manager. The user credentials for all users who login to each applications must be stored in the specified registries. For more information on configuring user registries, see Selecting user registries for Cloud Manager and API Manager.

Roles
: Create roles and assign them to members of the Admin organization to specify which tasks members are authorized to perform. For more information on assigning roles, see Administering members and roles.

Role Defaults
: The Role Defaults for the Admin organization are not configurable, but you can customize the base roles for provider organizations and consumer organizations. For more information on customizing base roles, see Role Defaults overview.

Endpoints
: View the list of endpoints configured for API Manager, provider APIs, and consumer APIs during the installation process.

Notifications
: Select the email server for your on-premises cloud notifications, configure notification settings, and specify the sender name and email address to be included in the emails. You can also preview and customize email templates used for notifications. For more information on setting up notifications, see Setting up notifications.
Note: Be sure to configure the email server, because it is needed for sending invitations and other notifications to users.

Catalog Defaults
: Specify one or more gateway services to be provided to every Catalog by default, so that Products (which contain APIs) can be made available to users. When a new Catalog is created in a provider organization, it is assigned to the default gateway service. Any changes you make to these settings do not affect existing Catalogs.
For more information on selecting gateway services, see Configuring the default gateway services for Catalogs.

Audit Setting
: Configure auditing settings to monitor API calls and log information about the calling users, the time of each call, and the activity involved in each event. The audit focuses on "who" did "what" and "when", including the request payload and the result of the operation. All create, update, and delete operations are audited. Generally, read operations are only audited if the calling user logged in with an OIDC Provider user registry. For more information about configuring the use of auditing, see Configuring audit logging to monitor user operations.

Use the following list to locate instructions for using specific cloud settings:

- **Change the name of your cloud**
  You can change the name of your cloud.
- **Configuring an email server for notifications**
  Connect to an email server in your API Connect on-premises cloud. The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service. Otherwise, the owner of the organization will not receive the email with the details that they require to access the API Manager.
- **Setting up notifications**
  Select an email server and set up the "sent from" information for notifications.
- **Configuring invitation timeouts**
  You can set the time period for when registration invitations for Cloud Settings access and the Admin organization expire.
- **Configuring API key settings**
  You can set the time period for when an API key expires, and optionally authorize it to be used multiple times.
- **Using the CLI to modify cloud settings**
  Use the toolkit CLI to edit the cloud settings.
- **Configuring Base64 encoding for temporary tokens**
  How to enable and disable Base64 encoding for temporary tokens.
- **Configuring the password-reset notification and timeout**
  Configure the time allowed for a user to reset a password, and customize the notification that the user receives.
- **Selecting user registries for Cloud Manager and API Manager**
  Select the user registries to authenticate users for Cloud Manager and API Manager.
- **Configuring timeouts for access tokens and refresh tokens**
  Configure timeout values for the access tokens and refresh tokens for user registries.
- **Viewing platform and UI endpoints**
  A view-only list of platform and UI endpoints is provided.
- **Configuring the default gateway services for Catalogs**
  Configure the list of default gateway services that are available to new Catalogs.
- **Managing the public/private key pairs for signing tokens**
  Manage the public/private key pairs that are used for signing tokens in API Connects.

Change the name of your cloud

You can change the name of your cloud.

About this task

You can change the name of your cloud using the Overview page.

One of the following roles is required to change the name of your cloud:

- Administrator
- Owner
- A custom role with the `cloud settings:manage`

Procedure

Follow these steps to change the name of your cloud:

1. In the Cloud Manager, click Settings.
2. Select Overview..
3. Click Edit to change the name of your cloud.
4. Enter a Title and the name is auto-generated. Enter a brief Summary and click Save.

Configuring an email server for notifications

Connect to an email server in your API Connect on-premises cloud. The email server sends registration invitations and other event-driven emails. You must configure the email server before you add any provider organizations or register a Portal service. Otherwise, the owner of the organization will not receive the email with the details that they require to access the API Manager.

Before you begin

You must complete the tasks in the following section:

1. Defining your Cloud Manager topology

About this task

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator
- Topology Administrator
- Custom role with **`Settings:Manage`** permission

The email server sends invitation emails with activation links and other system messages. More than one email server may be configured in Cloud Manager, but only one email server may be selected as the active server to send notifications at any one time. For instructions on how to select the active email server and to view a list of the email messages that are sent, see Setting up notifications.
Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

Procedure

To configure one or more email servers to use for email notifications in Cloud Manager, complete the following steps:

1. In the Cloud Manager, click Resources.
2. Choose Notifications.
3. Enter the values to configure the email server.

Field	Description
Title (required)	Enter a descriptive title for the email server. This title will display on the screen.
Name (required)	This field is auto-populated by the system and used as the internal field name.
Address (required)	IP address for the email server
Port (required)	Port that the email server runs on
Authenticate User	Enter a user name to use for authorization with the SMTP server. The user name is required if the SMTP server requires authentication.
Authenticate Password	Enter the password for the Authenticate User. This password is used for authorization with the SMTP server. The password is required if the SMTP server requires authentication.
TLS Client Profile (optional)	Select an optional TLS Client Profile that will be used to communicate with the email server if the Secure Connection option is selected.
Secure Connection	Select this option to specify that TLS security is used. The following conditions apply: • If this option is not selected then TLS security is not used even if the email server is configured to use the STARTTLS protocol. • If this option is selected and a TLS Client Profile is selected, that profile is used and the STARTTLS protocol configuration on the email server is honored. • If this option is selected and no TLS Client Profile is selected, the Default TLS client profile is used and the STARTTLS protocol configuration on the email server is honored.

4. You can send a test email to confirm that the email server is properly configured. Click Test email in the Test Connection panel to open a dialog window. Enter one or more valid email addresses in the Recipients field. Separate multiple recipients with commas. Click Send test email. Check that the test email was received by the recipients.
5. If the test email is successful, click Save to save the email server configuration. If the test is unsuccessful, check the configuration values and update as needed.

Note: In order to send emails, you must select an email server as your active server in Settings > Notifications.

Results

The email server is configured.

What to do next

You must now select the email server to use as your active email server in Cloud Settings > Notifications. You can enter the sender information, and customize the email templates. For more information, see Setting up notifications.
Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

Setting up notifications

Select an email server and set up the "sent from" information for notifications.

Before you begin

You must configure an email server to use for your notifications in Resources > Notifications, before you can set up the notification configuration in Settings > Notifications. See Configuring an email server for notifications for more information.

About this task

In Settings > Notifications, you select the email server that you want to use as the active server to send notification emails. An active email server is required to send email notifications. More than one email server may be configured in Cloud Manager, but only one email server may be selected to send notifications at any one time. The email server must be configured and selected for notifications before adding a Portal Service to your cloud.

You also configure the sender information to be included on all the emails and optionally, edit the standard text for the email templates.

One of the following roles is required to add roles to select the email server, enter the sender information, and edit the email templates:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` permission

Note: If you do not select an email server for sending notification emails, it is not possible for a user to reset their password from either the Cloud Manager or API Manager login pages.

Procedure

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Notifications.
3. To configure the active email server:
   a. Click Edit in the Sender & Email Server section.
   b. Select an email server in the list; a check mark is displayed next to the currently selected server.
   c. Click Save.
4. To delete an email server from the list, click the options menu next to the server name and select Delete.
5. To change the sender name and email address for notifications:
   a. Click Edit in the Sender & Email Server section.
   b. Fill in the Name and Email address that notifications will be sent from.
   c. Click Save.
6. To preview and edit the notification email templates, see Customizing email notification templates

What to do next

You can now create a provider organization.

- **Customizing email notification templates**
  API Connect includes a set of email templates for invitations and other notification emails. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

Customizing email notification templates

API Connect includes a set of email templates for invitations and other notification emails. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

About this task

The API Connect emails are sent automatically when certain system events occur. The email templates are organized by scope, as explained in Table 1.

Table 1. Notification scopes

Scope	Events that trigger the notification
admin	Admin organization activities such as invitations, portal sign up, and password reset.
catalog	Activities related to Catalogs; for example: such as application life cycle events, invitations to catalogs, product approvals, and subscription approvals.
consumer	Activities to consumer applications and subscription requests in the Developer Portal, invitations to Consumer organizations, and password reset requests for Developer Portal accounts.
provider	Invitations
space	Invitations

Email notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see Configuring sender details for email notifications in the Managing your APIs section.

One of the following roles is required to configure email notifications:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` permission

Note: You can also customize notifications by using the developer toolkit CLI, or by using the API Connect REST APIs. See Using the CLI to modify cloud settings, or API Connect REST APIs, for more information.
If you want to configure the notifications at the Provider Organization level, see Configuring notifications in the Managing your APIs section.

Procedure

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Notifications.
   On the Notifications page, templates display in a list, sorted by scope. When selecting a template to modify, be sure to select the version that applies to the appropriate scope.
3. To preview the text for a template, select Preview from the actions menu next to the template name.
4. To edit the text for the template, either click the template name, or select Edit in the action menu.
   When a template is opened, an attempt is made to get the language from the browser setting, but if this isn't possible the template defaults to English.

   The text includes variables; for example, `{{catalog}}`. The notification text is based on Handlebars syntax. Most variables are enclosed in double curly braces `{{ }}`, but can be enclosed in triple curly braces `{{{ }}}` to disable HTML escaping, when the variable is a URL link for example. For more information on Handlebars, see https://handlebarsjs.com/.

   To obtain the complete list of variables that are available for a particular notification template, complete the following steps:
   a. Log in to the management server from the command line as a member of the cloud administration organization; for details, see Logging in to a management server. You can use the same management server URL, user name, and password in the login command that you use to log in to the Cloud Manager user interface.
   b. Enter the following command:

   ```
   apic notification-templates:get template_name --server mgmt_endpoint_url --scope cloud --subcollection template_scope
   --fields variables --output -
   ```

   where:
   - *template_name* is the name of the required notification template, as displayed in the Template column in the user interface.
   - *template_scope* is the scope name displayed in the Scope column alongside that template.

   For example:

   ```
   apic notification-templates:get member-invitation --server https://myserver.com --scope cloud --subcollection catalog
   --fields variables --output -
   ```

   The variables that are available for the template are displayed, for example:

   ```
   variables:
     - org
     - catalog
     - activationLink
     - expiresAt
     - originator
     - originatorFirstName
     - originatorLastName
     - originatorEmail
     - username
     - email
     - firstName
     - lastName
   ```

   The `--output -` parameter causes the command output to be written to the command line. You can specify `--output filepath` to have the output written to a .yaml file at the specified location, or omit it altogether to have a file written to the current folder.
5. To view and edit the notification template in a different language, select one of the following supported languages from the View template in drop-down list:
   - Chinese (Simplified)
   - Chinese (Traditional)
   - Czech
   - Dutch
   - English (US English)
   - French
   - German
   - Italian
   - Japanese

- Korean
- Polish
- Portuguese
- Russian
- Spanish
- Turkish

6. Edit the Subject as required.
7. Select the Content type that you want to use for the template, from HTML, PlainText, or Both. The default content type is PlainText.
   An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
8. Edit the body of the template as required.
   For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 2. List of allowed HTML tags and their attributes

HTML tag	Attribute
`<a>`	`"class", "href", "hreflang", "style"`
`<em>`	`"class", "style"`
`<strong>`	`"class", "style"`
`<cite>`	`"class", "style"`
`<blockquote>`	`"class", "cite", "style"`
`<code>`	`"class", "style"`
`<ul>`	`"class", "type", "style"`
`<ol>`	`"class", "start", "type", "style"`
`<li>`	`"class", "style"`
`<dl>`	`"class", "style"`
`<dt>`	`"class", "style"`
`<dd>`	`"class", "style"`
`<h1>`	`"class", "id", "style"`
`<h2>`	`"class", "id", "style"`
`<h3>`	`"class", "id", "style"`
`<h4>`	`"class", "id", "style"`
`<h5>`	`"class", "id", "style"`
`<h6>`	`"class", "id", "style"`
`<p>`	`"class", "style"`
`<div>`	`"class", "style"`
` `	`"class", "style"`
`<span>`	`"class", "style"`
`<img>`	`"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"`
`<table>`	`"class", "id", "style"`
`<tr>`	`"class", "id", "style"`
`<td>`	`"class", "id", "style"`

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.
Images can be used by adding a **`<img src="https://path/to/image.png"/>`** tag in the template. The **`src`** attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

9. Click Save when done.
   Note: Edits made to a template are saved only for the specific language version that is edited.

Related information

- Configuring notifications

Configuring invitation timeouts

You can set the time period for when registration invitations for Cloud Settings access and the Admin organization expire.

About this task

One of the following roles is required to change the name of your cloud:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

Procedure

Follow these steps to set the invitation timeout:

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the invitation timeout setting that you want to change:
   - Cloud Settings Invitation Timeout
   - Admin Organization Invitation Timeout
4. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.

5. Click Save.

Results

Invitations contain a link, which expires after the specified the timeout. If the user does not register before the expiration, you can send a new invitation.

Configuring API key settings

You can set the time period for when an API key expires, and optionally authorize it to be used multiple times.

About this task

Applications exchange the API key exchange the API key for an access token.

Procedure

Follow these steps to set the API key timeout:

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the API Key timeout setting.
4. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
5. Select Multiple use of apikey to allow applications to exchange the API key for an access token multiple times.
   By default, the API key can be used only once.

   To enable this setting with the CLI, set the **api_key_multiple_uses** field to **true**.

6. Click Save.

Results

API Key times out based on the set timeout value.

Using the CLI to modify cloud settings

Use the toolkit CLI to edit the cloud settings.

About this task

Some cloud settings can be modified in the Cloud Manager UI, but all settings can be modified with the toolkit CLI. You can modify the cloud settings configuration file and upload the changes, or issue commands to modify individual settings directly. If multiple people modify the cloud settings, any newer settings overwrite older settings regardless of the method used to update the settings.

For more information about working with the toolkit CLI, see the following topics:

- For details on using **apic** to modify cloud settings, see apic cloud-settings:update.
- For details on all the cloud settings commands, see apic cloud-settings

Procedure

1. Run the following command to log in to a management server as a member of the cloud administration organization:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
   ```

   Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
   For more information on logging in, see Logging in to a management server.

2. Optionally retrieve the current list of cloud settings with the following command:

   ```
   apic cloud-settings:get --server mgmt_endpoint_url --output -
   ```

   where *mgmt_endpoint_url* is the URL of the server where the cloud settings are stored.

3. Add or modify settings using one of the following methods:
   - Issue each setting from the command line.
     For simple changes, you can issue commands to modify settings without editing the configuration file.
     - Issue the following command to start the update process:

       ```
       apic cloud-settings:update --server mgmt_endpoint_url -
       ```

       where *mgmt_endpoint_url* is the URL of the server where the cloud settings are stored. Include the hyphen at the end of the command to indicate that input follows on the next command line.

     - Type the information for a single setting on one line, and press ENTER to submit it.
       Specify each setting on a single line and use the following format to specify the setting's name and value:

       ```
       setting_name: setting_value
       ```

       For example, the following command updates a single setting:

       ```
       apic cloud-settings:update --server mgmt_endpoint_url -
       refresh_expires_in: expiration time
       Ctrl+D
       ```

     - Type Ctrl+D to indicate that the data entry is complete, as shown in the example.
   - Upload a file containing the settings.
     If you have multiple settings or lengthy values to add or modify, you might prefer to make all of the changes at once by creating or modifying the configuration file and then uploading it.
     - Create a .yaml file with your settings.
       You can choose the file name but must use YAML format. The file can contain as few or as many settings as needed.

       Settings use the following format; notice that the setting's name must be enclosed in quotation marks as shown in the example. The last setting in the file does not end with a comma.

       ```
       "setting_name": setting_value,
       ```

     - Upload the file with the following command:

       ```
       apic cloud-settings:update --server mgmt_endpoint_url my_cloud_settings.yaml
       ```

     For information on using input files with the toolkit CLI, see ../com.ibm.apic.toolkit.doc/rapic_cli_command_line_input.html.

4. When you are ready to log out of the management server, use the following command:

   ```
   apic logout --server mgmt_endpoint_url
   ```

Results

The settings are cached for a maximum of five minutes, so you might experience a delay before your changes take effect.

Configuring Base64 encoding for temporary tokens

How to enable and disable Base64 encoding for temporary tokens.

About this task

You can enable or disable Base64 encoding for temporary tokens. Temporary tokens are invitations, and password reset tokens. When Base64 encoding is enabled, the temporary token is encoded in Base64 format. When Base64 encoding is disabled, the temporary token remains in the JSON Web Token (JWT) format, and means that the length of the URL that the user receives is reduced.

Base64 encoding is enabled by default.

One of the following roles is required to configure Base64 encoding:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

Procedure

Follow these steps to configure Base64 encoding for temporary tokens:

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Onboarding.
3. Click Edit next to the Base64 encoding setting.
4. Select Enable to enable Base64 encoding, or Disable to keep temporary tokens in the JSON Web Token (JWT) format.
5. Click Save.

Results

You have successfully enabled or disabled Base64 encoding for temporary tokens.

Configuring the password-reset notification and timeout

Configure the time allowed for a user to reset a password, and customize the notification that the user receives.

Before you begin

Configure an email server that can be used for mailing notifications to users. If you do not configure an email server, a link cannot be send to a user who wants to reset a password. For information on configuring the email server, see Setting up notifications.

About this task

When a user clicks Forgot Password? and provides an email address on the sign-in page, an email is generated with a link where the user can reset the password. When you deploy API Connect, the password-reset settings are configured with default values. You can optionally modify the time-out setting to set a new expiration for resets as well as the notification that is emailed to users.

Procedure

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Onboarding.
3. Configure the password reset timeout:
   a. In the Settings navigation list, click Onboarding.
   b. Click Edit next to Reset Password Time to Live.
   c. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
   d. Click Save.
4. Customize the notification template for password resets:
   a. In the Settings navigation list, click Notifications.
   b. In the Notifications Templates list, select the appropriate version of the "password-reset" template by clicking the name.
      You can create different versions of the template for different user roles.
   c. Customize the template as explained in Customizing email notification templates.
   d. Click Save.

Results

Password reset notifications contain a link, which expires after the specified the timeout. If the user does not reset the password before the link expires, they can request a new link.

Selecting user registries for Cloud Manager and API Manager

Select the user registries to authenticate users for Cloud Manager and API Manager.

About this task

In Settings > User registries, you select one or more user registries to authenticate users of Cloud Manager and API Manager. The user credentials for all users who login to the applications must be stored in the selected registries.

One of the following roles is required to select the user registries for Cloud Manager and API Manager:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

For details on how to control which user registries can be used by provider organizations for authenticating Developer Portal users, or for securing APIs, see Setting visibility for a user registry.

Procedure

To select the user registries for Cloud Manager and API Manager, follow these steps:

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click User Registries.
3. The current list of user registries is displayed for Cloud Manager and for API Manager. To add user registries, click Edit in either the Cloud Manager or API Manager panel.
   Note: The list of user registries is derived from the user registries that were previously configured as a Resource in Resources > User Registries. For more information, see User registries overview
4. Select the user registry by placing a check mark next to the name in the list. Remove a user registry by removing the check mark.

5. Click Save.

Results

The user registry list is updated to reflect the selected user registries available for authenticating users.

What to do next

To set a user registry as the default, click the options menu next to the registry name in the list, and select Set as default.

- **Setting a default user registry for Cloud Manager and API Manager**
  You can select a default user registry for Cloud Manager and and for API Manager.

Related tasks

- Configuring an LDAP user registry in the Cloud Manager
- Configuring an Authentication URL user registry
- Configuring a Local User Registry

Setting a default user registry for Cloud Manager and API Manager

You can select a default user registry for Cloud Manager and and for API Manager.

About this task

In Settings > User registries, you can set a user registry as the default user registry.

One of the following roles is required to select the user registries for Cloud Manager and API Manager:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

Procedure

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click User Registries.
3. To set a user registry as the default, click the options menu next to the registry name in the list, and select Set as default.
   Note: The list of user registries is derived from the user registries that were previously configured as resources in Resources > User Registries. For more information, see User registries overview

Results

The user registry list is updated to reflect the default user registry.

Related tasks

- Configuring an LDAP user registry in the Cloud Manager
- Configuring an Authentication URL user registry
- Configuring a Local User Registry

Configuring timeouts for access tokens and refresh tokens

Configure timeout values for the access tokens and refresh tokens for user registries.

About this task

IBM® API Connect supports the use of access tokens and refresh tokens for user registries that authenticate users of the Cloud Manager, the API Manager, and the Developer Portal.

Access tokens allow users to sign in and remain logged in for a specified amount of time (the timeout period). Refresh tokens allow the access token to be renewed automatically without requiring the user to sign in again. Access tokens are configured by default, and you can modify the timeout setting. Refresh tokens are disabled by default but are configured with a default timeout when you enable them.

Refresh tokens, when enabled, are generated every time an access token is created. Refresh tokens can be used to obtain a new access token without having to prompt a user to re-login. If you enable the use of refresh tokens, then when the token expires it is refreshed automatically in the background so that the user can continue working without interruption. When the user logs out, the refresh token is revoked and the application can no longer use it.

Note: If users log in with non-OIDC user registries, the use of refresh tokens is supported in API Manager, API Designer, the Developer Portal, and the CLI Toolkit. When the refresh token expires, the user is always redirected to the login page.

If users log in with OIDC user registries:

- If the OIDC providers do not support refresh tokens, API Connect will not issue a refresh_token, regardless of the setting of **refresh_token_enabled** in the cloud settings.
- For more information about OIDC configuration, see Configuring an OIDC user registry in Cloud Manager.

You can configure access tokens and refresh tokens using the Cloud Manager UI, or the Toolkit CLI.

Using the UI to configure tokens

Configure timeout settings for access tokens and refresh tokens in the Cloud Manager Settings pages.

Procedure

1. Click in the navigation pane.
2. Configure the timeout for access tokens:
   a. On the Onboarding page, click Edit next to Access Token Time to Live.
   b. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
   c. Click Save.
3. Configure the timeout for refresh tokens:
   a. On the Onboarding page, click Edit next to Refresh Token Time to Live.
   b. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
   c. Click Save.

Using the CLI to configure tokens

Use the Toolkit CLI to configure timeout settings for access tokens and refresh tokens.

About this task

Use the following properties to specify refresh tokens for cloud settings:

- **access_token_expires_in**
  Integer. Represents the expiration time, in seconds, for access tokens issued by API Connect.
- **refresh_expires_in**
  Integer. Represents the expiration time, in seconds, for refresh tokens issued by API Connect. Must be greater than **access_token_expires_in**.
- **refresh_token_enabled**
  Boolean. Disabled by default. To enable, set to **true**. When enabled, generates a **refresh_token** field in the response to the **/token** API call.

The following extract from a cloud settings file shows example settings for these properties:

```
"type": "cloud_setting",
"api_version": "2.0.0",
"name": "cloud-setting",
"access_token_expires_in": 28800,
.
.
.
"refresh_expires_in": 57600,
"refresh_token_enabled": false,
.
.
.
```

Use the toolkit CLI (**apic**) to update the cloud settings with new values for the properties. You can either specify values as command line arguments, or enter them manually in a configuration file as explained in Using the CLI to modify cloud settings.

Viewing platform and UI endpoints

A view-only list of platform and UI endpoints is provided.

About this task

You can view the list of endpoints configured during the installation process. Included in this view-only list are the following endpoints:

- API Manager URL - the URL for opening the API Manager UI
- Platform REST API endpoint for the Admin and Provider REST APIs - this endpoint handles the REST APIs for Admin and Provider functions.
- Platform REST API endpoint for consumer REST APIs - this endpoint handles the REST APIs for consumer (Portal) functions.

One of the following roles is required to view endpoints:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` or `cloud-settings: view` permissions

Procedure

To view the endpoints, follow these steps:

1. In the Cloud Manager, click Settings.
2. Select Endpoints.
3. View the list of endpoints.

Configuring the default gateway services for Catalogs

Configure the list of default gateway services that are available to new Catalogs.

About this task

One of the following roles is required to configure Catalog Defaults:

- Administrator
- Owner
- A custom role with the `cloud settings:manage` permission

You publish APIs by adding them to a product and then publishing the product to a catalog. To be able to publish products to a catalog, the catalog must be assigned at least one gateway service so that the APIs in the product are available to be called at a gateway service endpoint. You can assign more than one gateway service to a catalog, and they can be of mixed types, DataPower® API Gateway and DataPower Gateway (v5 compatible).

A product is gateway type specific, either DataPower API Gateway or DataPower Gateway (v5 compatible). By default, when you publish a product to a catalog, it is published to all the assigned gateway services of the same gateway type as the product, but you can choose to publish the product only to selected gateway services, of that type, at publish time; see Publishing a draft product for more information. The APIs in the product will be available to be called at all the specified gateway service endpoints.

The procedure described in this topic specifies the gateway services that are assigned by default to newly created catalogs, but you can change the assigned gateway services for a specific catalog; see Creating and configuring catalogs.

Procedure

To configure the gateway services that are assigned to new Catalogs by default, complete the following steps:

1. In the Cloud Manager, click Settings.
2. In the Settings navigation list, click Catalog Defaults.
3. Click Edit to change the list of default gateway services.
4. Select the required gateway services, then click Save.

Results

When a new Catalog is created in a provider organization, the default gateway services are automatically assigned to it. Any changes you make to the list of default services affects only the Catalogs that are created after you save your changes. Existing Catalogs are not affected.

Related information

- Working with Catalogs
- API Connect gateway types

Managing the public/private key pairs for signing tokens

Manage the public/private key pairs that are used for signing tokens in API Connects.

API Connect uses public-key cryptography to secure tokens. By default, an API Connect deployment receives one public/private key pair, which is used for signing and verifying tokens. When a token is created, API Connect signs it with the deployment's private key. Any third-party can verify that digital signature with the deployment's public key.

The public/private key pair is generated for you by API Connect on the first start-up after installation, upgrade, or migration. The keys are stored in a keystore object.

Why do you need public/private key pairs?

API Connect uses a public/private key pair to sign and verify the following tokens:

- access_token: Verifies that the user has permission to access resources.
- id_token: Stored within the access token to specifies the user's ID.***Confirm?***
- temporary_token: Provides for short-lived access so that the user can log in and receive an access token. For example, when the user receives a link for resetting a password or to register for an account.

How does APIC use the public/private key pair?

Each API Connect deployment is assigned a public key, which is made openly available, plus a corresponding private key, which can only be accessed by that deployment. To verify a user's access to resources for a transaction, API Connect signs every token with the private key. Any other party can use the openly available public key to verify the signature. Because the private key is not shared, it can be used repeatedly instead of generating a new value for each signature.

When your app accesses an API endpoint, the access_token is in JWT format and includes a 'kid" value that is the new key.

Note: The signature for public/private key schema is larger than for shared secrets, so the access_token that is returned will also be larger. If you encounter issues with headers becoming too large, refer to the following topic: Resolving login problems by increasing HTTP header size.

How can you manage public/private key pairs?

By default, your deployment is assigned one key pair, which is stored in a keystore and never expires. The same key pair is used for signing and verify all tokens. The key pair consists of a public key and a private key:

- key_signing: The private key that API Connect uses for signing tokens
- key_verify: The public key that anyone can access and use for verifying token signatures

API Connect uses the following cloud settings to determine which public/private keystores are used for each type of token:

- access_token_keystore_urls: [*keystore_url_0, keystore_url_n*]
- id_token _keystore_urls: [*keystore_url_0, keystore_url_n*]
- temporary_token _keystore_urls: [*keystore_url_0, keystore_url_n*]

Each setting specifies an array of URLs where the keystores are hosted, with the keystore at position 0 representing the current key pair. You can optionally generate additional key pairs, store them in new keystores, and use them with the tokens to enhance security. Each key pair is stored in its own keystore and can be set to expire at a different time. Providing a different key pair for each type of token is a good practice because it limits the damage if a token is compromised.

If you specify one keystore for a token, that key pair is used for every new token of the same type, and it never expires. If you specify multiple keystores, you can assign a lifetime to each and when the keys in the first keystore expire, the keys are obtained from the next keystore in the list, and so on.

You can use the public/private key pairs and keystores to perform the following operations with tokens:

Sign a token
: The first keystore in the array is the current keystore; use its key_signing key to sign tokens.

Verify a token's signature
: To verify a token's signature, start with the current keystore (the first in the array). If the token doesn't match the key_verify key, move to the older keystores and compare the token with the key_verify value in each.

Add key pairs
: When you create a new key pair, add the keys to a separate keystore, and then add that keystore's URL to the end of the token's keystore array. The first keystore in the array is the "current" keystore, and subsequent keystores in the array are used in turn.

Revoke key pairs
: If you suspect that a token is compromised, you can revoke they keys for signing and verifying it. To revoke a key pair, delete the corresponding keystore from the token's keystore array. Removing a keystore removes the keys that were used for signing and verifying tokens, and invalidates all of the tokens that were issued with that key pair.
: Attention: You must always retain at least one key pair for signing and verifying tokens.

Rotate through keystores to periodically change the key pairs in use
: To rotate through key pairs for a token, configure at least two keystores for the token. Specify the "current" keystore first in the array, followed by the "next" keystores. Make sure that the each keystore's lifetime overlaps that of the next keystore so that keys from the current keystore don't expire before the new key pair takes effect.
: Before you configure the lifetime for a keystore, you can refer to the Onboarding section of the Cloud Settings page to quickly see the lifetime that is configured for each type of token. Then make sure you specify enough keystores, with long enough lifetimes, to span each token's lifetime. For example, if the temporary_token lives for 72 hours and the key used for signing it lives for 24 hours, you need to provide at least 3 sets of keys to span the token's lifetime:
:
: temporary_token: [keystore1, keystore2, keystore3]

- **Configuring public/private key pairs for tokens**
  Configure public/private key pairs for signing and verifying tokens in your API Connect deployment.

Configuring public/private key pairs for tokens

Configure public/private key pairs for signing and verifying tokens in your API Connect deployment.

About this task

By default, an API Connect deployment receives one public/private key pair, which is used repeatedly. The public/private key pair is generated for you the first time you start API Connect after installing or upgrading it. The keys are stored in a keystore object. You can add and revoke keys, and configure a rotation through keys for enhanced security.

You can configure key pairs for tokens in the Cloud Manager user interface, or by using the Toolkit CLI.

Using the Cloud Manager UI to configure public/private key pairs

Use the Cloud Manager user interface to configure key pairs for signing and verifying tokens.

About this task

Manage Keystore assignment and Keystore history for tokens

Procedure

1. In the Cloud Manager, click .
2. In the Settings navigation list, click Signed Token.

The Signed Token page displays the list of supported tokens for signing.

3. Review the current token settings:
   - Click Payload to display the payload code for the set of tokens.
   - Click History log next to a token to view its previous keystore assignments (this history never includes the current keystore assignment). The most recent assignment displays at the top of the list. You can remove any keystore assignment from the history by clicking .
4. To add or change a keystore assignment for a token, complete the following steps:
   a. On the Signed Token page, click Edit.
   b. On the Edit signed token page, select a keystore for the token you want to modify.
   c. Click Save to save your changes and update the keystore history.
   Changes take effect immediately.

   The previously selected keystore is added to the history for the token that you just updated. The history never includes the current keystore assignment.

Using the Toolkit CLI to configure public/private key pairs

Use the Toolkit CLI to configure key pairs for signing and verifying tokens.

About this task

Use the following properties to the keystore URLs that contain the public/private key pair for each type of token. For each setting, the list of keystores is an array of strings, with each string representing the complete URL of a keystore. The first keystore in the array (*keystore_url_0*) is the one that is currently in use.

- access_token:

  `access_token_keystore_urls: [`*`keystore_url_0, keystore_url_n`*`]`

- id_token:

  `id_token_keystore_urls: [`*`keystore_url_0, keystore_url_n`*`]`

- temporary_token:

  `temporary_token_keystore_urls: [`*`keystore_url_0, keystore_url_n`*`]`

The keystore URLs can be lengthy. You can avoid errors by writing the settings in a file and uploading that file. Format the settings as a .yaml file (you can choose the file name) and upload the file using the toolkit CLI. The following example configures a different keystore for each type of token:

```
access_token_keystore_urls:
  - >-
    https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/de5298dd-a00b-45a0-9d08-96c5d93e1273
id_token_keystore_urls:
  - >-
    https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/85a21745-8ac3-499d-929a-2d017bade815
temporary_token_keystore_urls:
  - >-
    https://mgmt_endpoint_url/api/orgs/1cdb71a1-b53c-4b66-b162-1b0f45ca8a54/keystores/63d6462f-c93d-439b-b422-0e604381e97a
```

Procedure

1. Create a YAML file with the keystore settings for each type of token as shown in the example.
   For information on using input files with the toolkit CLI, see Reading input from the command line.

2. Run the following command to log in to a management server as a member of the cloud administration organization:

   `apic login --server `*`mgmt_endpoint_url`***` --username `***`user_id`***` --password `***`password`***` --realm admin/`***`identity_provider`**

   Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

   You can determine which identity provider to use in the **`--realm`** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   `apic identity-providers:list --scope admin --server `*`mgmt_endpoint_url`***` --fields title,realm`**

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The **`title`** value should enable you to determine which identity provider to use; you can then copy the corresponding **`--realm`** parameter directly from the displayed **`realm`** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **`default-idp-1`**.
   For more information on logging in, see Logging in to a management server.

3. Upload the YAML file with the new settings by running the following command:

   `apic cloud-settings:update --server `*`mgmt_endpoint_url my_token_keystore_settings`***`.yaml`**

4. Run the following command to log out of the management server:

   `apic logout --server `*`mgmt_endpoint_url`**

Configuring API governance in the Cloud Manager

How to add API governance rulesets in the Cloud Manager, to validate and enforce organizational governance policies and best practices across your API development process.

Before you begin

Before you can create API governance rulesets, the API governance optional add-on must be enabled on your management subsystem by your system administrator. See Enabling API governance on Kubernetes, and Enabling API governance on VMware for more information. If API governance is enabled in your deployment, the API governance resource is displayed on the Resources page in the Cloud Manager.

The API governance service is available to all user roles.

About this task

API governance is an optional add-on to IBM® API Connect that can be used to validate and enforce organizational governance policies and best practices to your API development process. You configure API governance by creating one or more custom rulesets that contain a collection of rules that can then be used to check Swagger, OpenAPI, and AsyncAPI documents. API governance contains the following types of rulesets:

- Global rulesets - these are custom rulesets that you can keep private, or you can make available to some or all of your provider organizations.
- Default rulesets - these are default rulesets that are built-in to API Connect, and contain pre-configured rules that are available to all of your provider organizations, and cannot be edited.

API governance in IBM API Connect is based on the open source Spectral linter; for more information about Spectral, see https://docs.stoplight.io/docs/spectral/674b27b261c3c-overview.

Note:

- API governance in IBM API Connect only supports the creation of custom rulesets that contain rules that use the built-in Spectral core functions, as defined in https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions. The use of custom functions, for example rules that use functions that you have created yourself in JavaScript files, is not supported.
- Some of the Spectral rules within the Default rulesets contain the property `recommended: false`, which means that those rules are ignored during validation. However, if you create a new ruleset from one of these rulesets by using the Save as new ruleset option, the `recommended` property isn't transferred to the new ruleset. Therefore all of the rules will be used in the validation, unless you delete those rules from the ruleset. The Spectral ruleset names are prefixed by spectral-.
- You can also configure API governance rulesets directly in a provider organization. For more information, see Configuring API governance in the API Manager.

You can configure API governance rulesets by using the following methods:

- Create a new ruleset using the UI
- Import a ruleset
- Create a ruleset using the CLI

Procedure

- To create a new ruleset using the Cloud Manager UI, complete the following steps.
  1. In the Cloud Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. In the Global rulesets section, click Add > Create from scratch.
     The Create ruleset wizard opens.
  4. Provide the following Ruleset info details.

Property label	Required	Description	Data type
Title	Yes	A short descriptive name for the ruleset. This name is displayed in the list of global rulesets.	String
Name	Yes	An auto-generated, sluggified, string that is based on the title. It is not editable.	String
Version	Yes	The version number defaults to 1.0.0, but can be edited. You can have rulesets with the same name, but with different version numbers. The version number must be in the format `major.minor.patch`.	String
Description	No	An optional description of the ruleset.	String

  5. Click Next, and add an initial rule to your ruleset.
  6. Provide the following Rule info details.

Property label	Required	Description	Data type
Title	Yes	A short descriptive name for the rule.	String
Name	Yes	A short machine name for the rule. Can contain only alpha-numeric characters, underscores (_), and hyphens (-).	String
Description	Yes	A description of the rule.	String
Message	No	Provide a message to help users understand what the goal of the rule is. If no message is provided, the value of the Description property is used for the message output of the validation scan.	String
Severity	Yes	Select a severity level from the following options: • error • warn • info • hint • off error is selected by default.	String

  7. In the Given section, specify one or more values for the parts of the API document to target. To add more values, click Add. Values must be written in JSONPath.

8. In the Then section, specify the function type and options that will be used to evaluate the target values in the API document. Write this section in YAML syntax.
   In the following example, for a Given value of **$** (which means root level of the document), the Then section uses the Spectral core **truthy** function to check that the **info.contact** property value is not **false**, **""**, **0**, **null**, or **undefined**.

   ```
   - field: info.contact
     function: truthy
   ```

   The Then section can also contain a list of functions to be applied to the Given part of the document. In the following example, for a Given value of **$.info.version** (which refers to the **version** property inside the **info** property at the root level of the document), the Then section uses both the **truthy** function, and the Spectral core **pattern** function that checks that the **info.value** matches the regex expression that is given by **functionOptions.match**.

   ```
   - function: truthy
   - function: pattern
     functionOptions:
       match: "^[0-9]+.[0-9]+.[0-9]+(-[a-z0-9+.-]+)?"
   ```

   For a complete list of the Spectral core functions that you can use in your custom rules, along with examples of these functions, see https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions.
9. Click Create to create your draft ruleset.
   Your draft ruleset opens in design form view. Note that you can switch to the OpenAPI YAML source view by clicking the Source icon </>. To return to the design form view, click the Form icon .
10. You now have the following options:
    - You can continue to edit your ruleset. For example, you can add more rules by clicking the Create rule icon ⊕ alongside Rules in the navigation pane. Remember to click Save when you finish editing.
    - If you don't want to edit your ruleset further now, click API governance in the breadcrumb trail to return to the API governance overview page.

- To import an existing ruleset, complete the following steps. Remember that the rules in your imported ruleset can contain only built-in Spectral core functions, as defined in https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions.
  1. In the Cloud Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. In the Global rulesets section, click Add > Import.
     The Import ruleset wizard opens.
  4. Drag a file into the Upload file box, or click in the Upload file box to select a file to upload. The maximum file size is 100 kb, and the supported file types are YAML, YML, and JSON.
  5. Click Next.
  6. Edit the Ruleset info details as required, and click Import.
     Your draft ruleset opens in design form view. Note that you can switch to the OpenAPI YAML source view by clicking the Source icon </>. To return to the design form view, click the Form icon .
  7. You now have the following options:
     - You can continue to edit your ruleset. For example, you can add more rules by clicking the Create rule icon ⊕ alongside Rules in the navigation pane. Remember to click Save when you finish editing.
     - If you don't want to edit your ruleset further now, click API governance in the breadcrumb trail to return to the API governance overview page.
- To create a ruleset using the API Connect toolkit CLI, complete the following steps.
  1. Create a JSON file containing the new ruleset.
     The following example shows the contents of a Spectral Ruleset file for API Manager:

     ```
     {
       "name": "custom-ruleset",
       "title": "custom-ruleset",
       "description": "Info validation",
       "ruleset_type": "global-custom",
       "ruleset_version": "1.0.0",
       "rules": [
         {
           "name": "contact",
           "severity": "error",
           "formats": [
             "oas2",
             "oas3"
           ],
           "given": [
             "$"
           ],
           "description": "The info object must contain either a contact email or contact url",
           "then": [{
             "field": "info.contact",
             "function": "truthy"
           }]
         },
         {
           "name": "description",
           "severity": "error",
           "formats": [
             "oas2",
             "oas3"
           ],
           "given": [
             "$"
           ],
           "description": "The info object must contain a description",
           "then": [{
             "field": "info.description",
             "function": "truthy"
           }]
     ```

```
    }]
}
```

2. Log in to the toolkit as an Admin.
3. Run the `rulesets:create` toolkit command in `governance` mode:

   ```
   apic -m governance rulesets:create --org <target_org> --server <server_url> RULESET_FILE
   ```

   where:
   - `<target_org>` is the name of the organization that owns the new ruleset.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `RULESET_FILE` is the path and filename of the JSON file that contains the ruleset.

Results

Your draft ruleset is now listed in the table of Global rulesets.

What to do next

You can validate the ruleset against an API document by clicking Validate, selecting the rules that you want to validate, and uploading an API file. The results of the validation are displayed in a scorecard.

When you finish editing your ruleset, you can publish it to your provider organizations. Click the options menu icon ⋮ either next to the ruleset that you want to publish, or from within the ruleset viewer. Select Publish, and then click Publish again to confirm. The status of your ruleset changes from Draft to Published, and can now be used by API developers to validate their API documents. For more information, see Validating an API document by using API governance.

Note:

- After a ruleset is published, the ruleset information and rules can no longer be edited. If you need to update this information, you must create a new version by clicking the options menu icon ⋮ either next to the ruleset that you want to edit, or from within the ruleset viewer, and selecting Save as new ruleset.
- By default, rulesets are published to all of your provider organizations. You must edit the visibility after publishing, if you want to change which provider organizations can use your rulesets.

If you want to change the visibility of your ruleset, click the options menu icon ⋮ either next to the ruleset that you want to edit, or from within the ruleset viewer. Then click Edit visibility, and select from the following visibility options:

- Private - the ruleset is not visible and cannot be used by any provider organization.
- Public - the ruleset is visible and can be used by all provider organizations.
- Custom - the ruleset is visible to and can be used by only the provider organizations that are designated by you.

Click Save to confirm your visibility option.

- **The API governance ruleset lifecycle in the Cloud Manager**
  When you manage your custom API governance rulesets in the Cloud Manager UI, you move them through a series of lifecycle states. From initially creating a draft ruleset, through to publishing to make the ruleset available to your API developers, to creating new versions, and to eventual archiving.
- **Updating rulesets and rules with the toolkit CLI**
  Use the API Connect toolkit CLI to update a ruleset or individual rules.

The API governance ruleset lifecycle in the Cloud Manager

When you manage your custom API governance rulesets in the Cloud Manager UI, you move them through a series of lifecycle states. From initially creating a draft ruleset, through to publishing to make the ruleset available to your API developers, to creating new versions, and to eventual archiving.

The following sections describe the various lifecycle states for a custom API governance ruleset.

Note:

- Approval is not required for any of the lifecycle state transitions.
- Default rulesets are built-in to API Connect, and contain pre-configured rules that are available to all of your provider organizations, and cannot be edited. These rulesets are always in the Published state, and their visibility cannot be changed.

Draft

The draft state for a ruleset is when the ruleset is not published, and isn't visible any provider organization. The ruleset can continue to be edited, and its Status is shown as Draft.

Published

When you publish a ruleset, it is made visible to all of your provider organizations, and cannot be changed. The ruleset Status is shown as Published, and the ruleset visibility is shown as Public. If you want to change which provider organizations can use your rulesets, you must edit the visibility after publishing. You can select from the following visibility options:

- Private - the ruleset is not visible and cannot be used by any provider organization.
- Public - the ruleset is visible and can be used by all provider organizations.
- Custom - the ruleset is visible to and can be used by only the provider organizations that are designated by you.

Any other changes require a new version of the ruleset to be created by using the Save as new ruleset option.

Archived

When you archive a ruleset, it becomes a ready-only, fixed version of the ruleset that cannot be used for validation by any provider organization. The ruleset Status is shown as Archived.
Note: If you cannot see your Archived ruleset, ensure that the Edit filters option on the ruleset table is set to include Archived.
You can restore archived rulesets by using the Restore option, after which the Status returns to Published.

Also, you can delete Archived rulesets by using the Delete option.

Updating rulesets and rules with the toolkit CLI

Use the API Connect toolkit CLI to update a ruleset or individual rules.

Updating a ruleset

1. Create a JSON file that defines the updated ruleset.
   The following example shows an updated ruleset:

   ```
   {
       "title": "custom-ruleset-2",
       "description": "short ruleset updated",
       "ruleset_type": "custom"
   }
   ```

   where:
   - `title` is updated to a new title for the ruleset.
   - `description` is updated to a new description for the ruleset.
   - `ruleset_type` is updated to be custom for the ruleset.
2. Log in to the toolkit as an Admin or as the owner of the provider organization that owns the ruleset.
3. Run the `rulesets:update` toolkit command in `governance` mode:

   ```
   apic -m governance rulesets:update --org <target_org> --server <server_url> RULESET RULESET_FILE
   ```

   where:
   - `<target_org>` is the name of the organization that owns the ruleset.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `RULESET_FILE` is the path and filename of the JSON file that contains the updated ruleset.
   - `RULESET` is the name or the unique ID of the updated ruleset.
     To determine the ID of a ruleset, run the following command:

     ```
     apic -m governance rulesets:list --org <target_org> --server <server_url>
     ```

Updating a rule

1. Create a JSON file that defines the updated rule.
   The following example shows an updated rule in a file called:

   ```
   {
     "description": "Updated the rule",
     "message": "Schema of type array must specify maxItems and minItems. {{path}} {{error}}",
     "formats": [
       "oas3"
     ],
     "severity": "warn",
     "recommended": true,
     "given": [
       "$..[?(@.type==\"array\")]"
     ]
   }
   ```

   where:
   - `description` is updated to a new title for the rule
   - `message` is updated to a new message for the rule
   - `formats` array is replaced to have only the format oas3 for the rule
   - `severity` is updated to the warn value for the rule
   - `recommended` is updated to have a new value for the rule
   - `given` array is replaced to have new values for the rule
2. Log in to the toolkit as an Admin or as the owner of the provider organization that owns the rule.
3. Run the `rules:update` toolkit command in `governance` mode.
   There are two ways to specify the updated rule in the command, depending on whether you provide the rule name and version, or the rule ID:

   Specify the rule name and version in the update command

   ```
   apic -m governance rules:update --org <target_org> --server <server_url> --ruleset <ruleset_name or ruleset_id>
   NAME:VERSION RULE_FILE
   ```

   where:
   - `<target_org>` is the name of the organization that owns the rule.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `<ruleset_name or ruleset_id>` is the name or the unique ID of the ruleset to which the updated rule belongs.

- **NAME** is the name of the updated rule.
- **VERSION** is the version of the updated rule; increase the version when you update the rule, using the **major.minor.patch** format as in "1.0.1".
- **RULE_FILE** is the path and filename of the JSON file that contains the updated rule.

Specify the rule ID in the update command

Instead of providing the rule's name and version, you can provide its unique ID instead by running the following command:

```
apic -m governance rules:update --org <target_org> --server <server_url> --ruleset <ruleset_name or ruleset_id> --id <rule_id> RULE_FILE
```

where:

- ***<target_org>*** is the name of the organization that owns the rule.
- ***<server_url>*** is the management server URL that you specified when you logged in to the toolkit.
- ***<ruleset_name or ruleset_id>*** is the name or the unique ID of the ruleset to which the update rule belongs.
- **RULE_FILE** is the path and filename of the JSON file that contains the updated rule.
- ***<rule_id>*** is the value of the updated rule's unique ID.
  To determine the ID of a rule, run the following command:

  ```
  apic -m governance rules:list --org <target_org> --server <server_url> --ruleset <ruleset_name> RULE_NAME
  ```

Administering provider organizations

Manage the provider organizations who have accounts to publish APIs in your API Connect cloud.

About this task

Perform these tasks to manage your provider organizations:

- **Creating a provider organization**
  For developers to publish APIs, they must be a member of a provider organization. The first step is to create a provider organization and specify an organization owner. Then members can be added to the provider organization and can start publishing APIs.
- **Editing a provider organization**
  You can edit the title of a provider organization, and also transfer ownership to a new organization owner.
- **Deleting a provider organization**
  You can delete a provider organization. All Catalogs and Spaces are also removed.
- **Viewing current provider organizations**
  View the list of current provider organizations for your cloud, including organization title, owner's name and email address, and status. From the list screen, you can add new provider organizations and owners, edit and delete provider organizations, send messages, and change owners.
- **Sending messages to provider organization owners**
  You can send an email message to the owner of a provider organization. You can also style the email to match your business theme.

Related tasks

- Creating a provider organization
- Editing a provider organization
- Deleting a provider organization
- Viewing current provider organizations

Creating a provider organization

For developers to publish APIs, they must be a member of a provider organization. The first step is to create a provider organization and specify an organization owner. Then members can be added to the provider organization and can start publishing APIs.

Before you begin

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

You must also complete the following tasks before beginning:

- Configuring an email server for notifications
- Setting up notifications

About this task

Note that an email server must be configured and an active email server must be selected before a provider organization account can be created.

There are two options for creating a provider organization account, as follows:

- Option 1: Create a provider organization and send an email invitation to a new user to register and become the organization owner. Choose the Add > Invite organization owner option.

- Option 2: Create a provider organization and specify the user who will be the organization owner. Choose the Add > Create organization option.

Procedure

- **Option 1:** To create a provider organization and invite a new user to register as the owner, complete the following steps:
  1. In the Cloud Manager, click Provider Organizations.
  2. Click Add > Invite organization owner
  3. Enter a valid email address and click Invite. An email containing an activation link is sent to the email address inviting the person to register. The link opens the registration form. Alternately, you can copy and paste the activation link into an email; click Get Activation Link and click Copy in the pop up window. Note that the email server requires the Authenticate User and Authenticate Password values in order to send invitations. See Configuring an email server for notifications.

     The new provider organization is placed in Pending status, and is changed to Enabled status after the owner completes the registration form to activate their account. The owner can then log in to the API Manager user interface with the API Connect user name that they specified during account activation.

     Note: If you use this option with a Local User Registry, and the user already exists in the registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. This stipulation includes users that were previously provider organization owners but whose organization was deleted.
- **Option 2:** To create a provider organization and specify the user who will be the organization owner, complete the followings steps:
  1. In the Cloud Manager, click Provider Organizations.
  2. Click Add > Create organization.
  3. In the Title field, enter a title for the provider organization. If desired, the title can be edited after the provider organization is created.
  4. The internal Name is automatically generated for you by the system. The Name field contains the string that is included in the organization segment of the URL for API calls. For more information, see Calling an API.
     Tip: The Name field is auto-generated to ensure its validity in the URL.
     The value in the Name field is also used to identify the provider organization in developer toolkit CLI commands. To view the CLI commands to manage provider organizations, see apic orgs.

  5. Select the user registry for the provider organization owner.
     The remaining procedure varies according to the type of the selected user registry, as follows:
     - Local User Registry
       - Select whether the user is an Existing user or a New User.
         - For an existing user, complete the following steps:
           - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
           - Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.
         - For a new user, complete the following steps:
           - Enter a unique user name for the new user.
           - Supply an email address, name details, and a password.
           - Click Create. The user account is created, and the provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface with the specified user name and password; the user interface open in the new provider organization.
     - LDAP
       - Enter the name of a user that exists in the selected user registry.
       - Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface with their LDAP user name. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.
     - Authentication URL and OIDC
       - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must use the invitation mechanism described in Option 1.
       - Click Create. The provider organization is created and immediately enabled. The specified owner can log in to the API Manager user interface. If the new provider organization is the user's only provider organization, the API Manager user interface opens in that provider organization; if the user is a member of more than one provider organization, they can then select the new provider organization from the Organization list.

What to do next

For details of how an invited organization owner activates their account see Activating your API Manager user account. For details of how to log in to the API Manager user interface, see Accessing the API Manager user interface.

Editing a provider organization

You can edit the title of a provider organization, and also transfer ownership to a new organization owner.

Before you begin

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

About this task

If you transfer ownership, the new organization owner can be an existing user, or you can create a new user. When the ownership changes, the new owner is assigned administration privileges for the organization, and the previous owner has their administration privileges removed. Optionally, you can also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner. If required, you can restore the privileges to the previous owner as described in Administering members and roles.

Procedure

To update a provider organization, complete the following tasks:

1. In the Cloud Manager, click Provider Organizations.
2. Click the name of the provider organization that you want to update.
3. To change the title, enter a new title in the Title field, then click Save.
4. To transfer ownership, select the user registry for the provider organization owner you want to transfer ownership to.
   The remaining procedure varies according to the type of the selected user registry, as follows:
   - Local User Registry
     - Select whether the user is an Existing user or a New User.
       - For an existing user, complete the following steps:
         - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
         - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
         - Click Save.
       - For a new user, complete the following steps:
         - Enter a unique user name for the new user.
         - Supply an email address, name details, and a password.
         - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
         - Click Save.
   - LDAP
     - Enter the name of a user that exists in the selected user registry.
     - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
     - Click Save.
   - Authentication URL and OIDC
     - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must ensure they have been invited to the organization first.
     - To also have the ownership of all Catalogs and Spaces in the organization transferred to the new organization owner, select Also transfer ownership of owned Catalogs and Spaces.
     - Click Save.

Deleting a provider organization

You can delete a provider organization. All Catalogs and Spaces are also removed.

Before you begin

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

About this task

The user account for the owner of the deleted provider organization remains in the associated API Connect user registry. The user can still log in to the API Manager user interface; if they are a member of one or more other provider organizations then those organizations are accessible, otherwise only information links are available, with no access to view or manage any resources.
The user can be added again as a provider organization owner at a later time; see Creating a provider organization.

Procedure

To delete a provider organization, complete the following tasks:

1. In the Cloud Manager, click Provider Organizations.
2. Navigate to the provider organization that you want to delete in the list and select Delete from the Actions menu.
3. Confirm that you want to delete the provider organization.
   The provider organization is deleted and removed from the Provider Organizations list.

Results

The provider organization, and its Catalogs and Spaces, are deleted.

Viewing current provider organizations

View the list of current provider organizations for your cloud, including organization title, owner's name and email address, and status. From the list screen, you can add new provider organizations and owners, edit and delete provider organizations, send messages, and change owners.

Before you begin

You must also complete the following tasks before beginning:

- Configuring an email server for notifications
- Setting up notifications
- Creating a provider organization

This task can be completed by users who are assigned one of the following roles:

- All roles
- Custom role with the `Provider-org:Manage` or the `Provider-org:View` permissions

About this task

All roles can view the list of provider organizations. Use the provider organization list screen to manage your provider organizations.

Procedure

1. In the Cloud Manager, click Provider Organizations.
2. View the list of provider organizations and their status. The statuses are as follows:
   - **Pending**: A provider organization is in `Pending` status until the invited owner successfully completes the registration form.
   - **Active**: A provider organization is in `Active` status if the owner is an existing user with an account in a user registry.
3. The Actions menu performs different functions depending on the organization's status.Use the Actions menu to edit or delete an Active organization. For Pending organizations, use the Actions menu to delete the organization and to resend the invitation to the owner
   - For an Enabled organization, you can Edit or Delete the organization.
   - For a Pending organization, you can Resend the Invitation, or Delete the request.
4. Add new provider organizations using the Add menu. See Creating a provider organization

Related tasks

- Creating a provider organization
- Editing a provider organization
- Deleting a provider organization

Sending messages to provider organization owners

You can send an email message to the owner of a provider organization. You can also style the email to match your business theme.

Before you begin

This task can be completed by users who are assigned the following roles:

- Cloud Owner
- Cloud Administrator
- Organization Manager
- Custom role with the `Provider-org:Manage` permission

Note: You can also send email messages by using the developer toolkit CLI, or by using the API Connect REST APIs. See Cloud administration commands, or API Connect REST APIs, for more information.

Procedure

To send a message, complete the following steps:

1. In the Cloud Manager, click Provider Organizations.
2. Click the options icon ⋮ alongside the provider organization that you want to work with, then click Message owner.
   The Send message window opens, and the email address of the provider organization owner is displayed as the recipient.
3. Enter the subject of the message in the Subject field.
4. Select the Content type that you want to use for the entering the content of the message, from HTML, PlainText, or Both. The default content type is PlainText.
   An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
   For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 1. List of allowed HTML tags and their attributes

HTML tag	Attribute
`<a>`	`"class", "href", "hreflang", "style"`
`<em>`	`"class", "style"`

HTML tag	Attribute
`<strong>`	`"class", "style"`
`<cite>`	`"class", "style"`
`<blockquote>`	`"class", "cite", "style"`
`<code>`	`"class", "style"`
`<ul>`	`"class", "type", "style"`
`<ol>`	`"class", "start", "type", "style"`
`<li>`	`"class", "style"`
`<dl>`	`"class", "style"`
`<dt>`	`"class", "style"`
`<dd>`	`"class", "style"`
`<h1>`	`"class", "id", "style"`
`<h2>`	`"class", "id", "style"`
`<h3>`	`"class", "id", "style"`
`<h4>`	`"class", "id", "style"`
`<h5>`	`"class", "id", "style"`
`<h6>`	`"class", "id", "style"`
`<p>`	`"class", "style"`
`<div>`	`"class", "style"`
` `	`"class", "style"`
`<span>`	`"class", "style"`
`<img>`	`"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"`
`<table>`	`"class", "id", "style"`
`<tr>`	`"class", "id", "style"`
`<td>`	`"class", "id", "style"`

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.
Images can be used by adding a **`<img src="https://path/to/image.png"/>`** tag in the template. The **`src`** attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

5. Enter the content of the message.
6. Click Send.

Administering members and roles

Cloud Administrators can add members and assign them roles to enable them to work in Cloud Manager. The Cloud and Topology Administrators can also create roles and role defaults. You can also delete users to prevent them from accessing Cloud Manager.

About this task

To manage roles, add members, and assign roles to member, perform the following tasks:

- **Creating roles in the admin organization**
  Cloud Manager ships with a set of pre-configured roles. As the Cloud Manager administrator, you can create custom roles, add permissions, and assign the roles to members of the Admin organization.
- **Editing roles in the admin organization**
  As the Cloud Manager administrator, you can edit the roles in the Admin Organization. Both pre-configured roles and custom roles may be edited, except for the Owner, Viewer, and Member roles. The Owner, Viewer, and Member roles may not be edited or deleted.
- **Configuring LDAP group mappings on Cloud Manager user roles**
  As a Cloud Manager administrator, you can configure LDAP group mapping on API Connect roles in the Admin organization by using the developer toolkit CLI.
- **Deleting roles in the admin organization**
  Roles can be deleted (except for the Owner, Viewer, and Member roles) from the Admin organization. If there are members assigned to a role that has been deleted, they will be assigned the Viewer role.
- **Viewing members and roles**
  Use the Members page to view the current members of the Admin organization and manage the role assignments.
- **Adding members to the admin organization**
  As the Cloud Manager administrator, you can add and remove members of the Admin organization and assign roles to them. Once removed, a user can no longer access the Cloud Manager.
- **Assigning roles to members**
  As the Cloud Manager administrator, you manage the roles that are assigned to members. A member's role determines the tasks they are authorized to perform in Cloud Manager.
- **Deleting a member**
  As the Cloud Manager administrator, you can delete members of the API Connect cloud administration organization. Once deleted, the user and associated roles are removed, however the user's account still remains in API Connect.
- **Role Defaults overview**
  Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

Creating roles in the admin organization

Cloud Manager ships with a set of pre-configured roles. As the Cloud Manager administrator, you can create custom roles, add permissions, and assign the roles to members of the Admin organization.

About this task

If you are assigned a role that has the `Settings: manage` permission, you can create custom roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.
One of the following roles is required to add custom roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

Procedure

Perform the following steps to add custom roles to the Admin Organization:

1. In the Cloud Manager, click Settings.
2. Select Roles.
3. Click Add to add a new role.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see User roles and permissions in the Cloud Manager UI.
6. Save the custom role.

Results

The custom role appears in the list of roles and can be assigned to members.

What to do next

Assign the role to a member in the Members list. See Assigning roles to members

Related tasks

- Editing roles in the admin organization

Editing roles in the admin organization

As the Cloud Manager administrator, you can edit the roles in the Admin Organization. Both pre-configured roles and custom roles may be edited, except for the Owner, Viewer, and Member roles. The Owner, Viewer, and Member roles may not be edited or deleted.

About this task

You can customize the per-configured roles (except for the Owner, Viewer, and Member roles) to reflect your business needs.

If you are assigned a role that has the `Settings: manage` permission, you can edit the roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

Following are the edits you can perform for a role:

- Change the title
- Add or delete permissions

Procedure

Perform the following steps to edit roles in the Admin organization:

1. In the Cloud Manager, click Settings.
2. Select Roles.
3. Locate the Role you want to edit, and select Edit from the overflow menu.
4. Enter the desired Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see User roles and permissions in the Cloud Manager UI.
6. Save the updated role.

Results

The Role will be updated with a new name and/or permissions. An updated role will affect the members who are assigned the role. If you change the name of a role, you will have to assign it to the members. If you change the permissions for a role, the way that members use the product may change.

Configuring LDAP group mappings on Cloud Manager user roles

As a Cloud Manager administrator, you can configure LDAP group mapping on API Connect roles in the Admin organization by using the developer toolkit CLI.

Before you begin

You must have an LDAP user registry resource in the Cloud Manager that has the **`external_group_mapping_enabled`** configuration set to **`true`**. See Using the CLI to configure a shared LDAP user registry for information.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

Note: The Admin organization roles apply only to Cloud Manager users. If you want to apply LDAP group mapping to API Manager users, see Configuring LDAP group mapping on API Manager user roles. LDAP group mapping cannot be applied to Developer Portal users.

About this task

You can map external LDAP groups to the API Connect preconfigured user roles (except for the Owner and Member roles), as well as to any custom user roles, to reflect your business needs.

Notes:

- After LDAP group mapping is enabled on a role, user onboarding always honors the group mappings.
- Once on-boarded, user membership in API Connect is valid throughout the login period (**`access_token_ttl`**), irrespective of any changes in the external LDAP registry. Membership is updated only on the next login, when the LDAP information is fetched and refreshed.
- One or more API Connect roles can be mapped to one or more LDAP groups, and one or more LDAP groups can be mapped to a role.
- When multiple LDAP groups are mapped to a single role, it means that a user from any one of the LDAP groups can log in to API Connect.
- If a user is removed from the external LDAP user registry, then to ensure quick removal from API Connect you must also delete the user membership in API Connect.

You can configure LDAP group mappings using one of the following methods:

- Using the UI to configure LDAP group mappings
- Using the CLI to configure LDAP group mappings

Using the UI to configure LDAP group mappings

Use the Cloud Manager UI to configure LDAP group mappings.

Procedure

1. Login to the Cloud Manager UI as a user with the appropriate permissions.
2. Click Settings > Roles > Add a new Role or edit existing role .
3. Select the following settings:
   - Enable external group mapping - select to enable external group mapping for this role.
   - LDAP user Registry - LDAP user registry list which has external group mapping enabled. User must select one of the LDAP user registry from this list.
   - LDAP group names - add one or more LDAP group names that you want to map to the user role.
   - User group prefix - add the prefix for the LDAP user group.
   - User group suffix - add the suffix for the LDAP user group.
4. Save your changes.

Results

The role is updated with the LDAP group mapping information. Users can now log on to the Cloud Manager, and automatically be assigned the correct access permissions for their role.

Using the CLI to configure LDAP group mappings

Use the toolkit CLI to configure LDAP group mappings.

About this task

For detailed information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Procedure

Perform the following steps to map external LDAP groups to Cloud Manager user roles.

1. Log in to the management server CLI.
   Before you can update the role configuration, you must log in to your management server from the developer toolkit CLI as a member of the cloud administration organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
```

where *mgmt_endpoint_url* is the platform API endpoint URL.
You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For full details of the login command, see Logging in to a management server.

2. Run the following command to get the URL of the LDAP user registry resource in the Cloud Manager that you want to map the user roles to:

```
apic user-registries:get  ldap_user_registry --org admin --server mgmt_endpoint_url --output -
```

where:
- *ldap_user_registry* is the name or ID of your LDAP user registry resource.
- **--org admin** means that the registry details are retrieved from the admin organization.
- *mgmt_endpoint_url* is the platform API endpoint URL.

This command outputs the configuration details of your LDAP user registry, and the **url:** is shown at the end of the list, for example:

```
type: user_registry
api_version: 2.0.0
id: 35e75bad-1d89-4a65-a70f-xxxxxx
name: ldap
title: LDAP
integration_url: >-
  https://server.com/api/cloud/integrations/user-registry/147b5fb1-e88e-41e3-90e9-xxxxxx
registry_type: ldap
user_managed: false
user_registry_managed: false
external_group_mapping_enabled: true
...
url: >-
  https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
```

3. Create a role yaml file that contains the following LDAP group mapping configuration properties:

```
external_group_mapping:
  user_registry_url: https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
  ldap_groups:
    - 'cn=apic-administrators,ou=ibmgroups,o=ibm.com'
    - 'cn=apic-developers,ou=ibmgroups,o=ibm.com'
  user_group_filter_prefix: (&(uniquemember=
  user_group_filter_suffix: ))(objectClass=groupOfUniqueNames))
```

Where:
- **user_registry_url** is the URL of your LDAP user registry resource from Step 2.
- **ldap_groups** is a list of the LDAP group names that you want to map to the user role.
- **user_group_filter_prefix** is the prefix for the LDAP user group.
- **user_group_filter_suffix** is the suffix for the LDAP user group.

4. Run the following command to update the user role with the **external_group_mapping** configuration properties:

```
apic roles:update role_name --scope org --org admin --server mgmt_endpoint_url mapping_properties_file
```

Where:
- *role_name* is the name of the user role that you want to add the LDAP group mapping to.
- **--scope** is the organization scope that you want the update to apply to. For this task, the value must be **org**.
- **--org admin** means that the user role is updated on the admin organization.
- *mgmt_endpoint_url* is the platform API endpoint URL.
- *mapping_properties_file* is the name of your mapping properties file from Step 3, for example **role_mapping_file.yaml**.

If you prefer to enter the configuration properties interactively on the command line, you can substitute the *mapping_properties_file* for a terminating hyphen character **-**, and enter the information manually, followed by pressing **CTRL D** to terminate the input.
If you want to create a custom role that includes LDAP group mapping, you can include the **external_group_mapping** configuration section in the **role_file**, and then create the new role by using the **apic roles:create** command.

For more information about the **apic roles** commands, see **apic roles** in the CLI reference section.

Results

The role is updated with the LDAP group mapping information. Users can now log on to the Cloud Manager, and automatically be assigned the correct access permissions for their role.

Deleting roles in the admin organization

Roles can be deleted (except for the Owner, Viewer, and Member roles) from the Admin organization. If there are members assigned to a role that has been deleted, they will be assigned the Viewer role.

About this task

All custom roles may be deleted. Pre-configured roles can also be deleted, except for the Owner, Viewer, and Member roles. Owner, viewer, and Member role cannot be deleted or edited. Members who are assigned a deleted role are given the most limited role, Viewer, with View-only privileges. These members will need to be assigned a new role if they require any other privileges.

If you are assigned a role that has the `Settings: manage` permission, you can delete roles for the Admin organization. The Admin organization roles apply only to Cloud Manager users.
One of the following roles is required to delete custom roles:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: manage` permissions

Procedure

Perform the following steps to delete roles in the Admin organization:

1. In the Cloud Manager, click Settings.
2. Select Roles.
3. Locate the Role you want to delete, and select Delete from the overflow menu.
4. Confirm that you want to proceed with deleting the role.

Results

The Role will be deleted. Members who had been assigned the role are given the Viewer role.

Viewing members and roles

Use the Members page to view the current members of the Admin organization and manage the role assignments.

Procedure

To view the members of the Admin organization and their roles in Cloud Manager, follow these steps:

1. In the Cloud Manager, click Members.
2. From the Members list, you can scroll through the list of users to see their current roles and their status.
   - **Pending** - The invitation with the activation link has been sent to the user, but they have not yet completed the registration form.
   - **Enabled** - The user has completed the registration form and is now able to login and access the features associated with their role.

   Note: If your role has the proper permissions, you can assign roles, add, and delete members from this list.

What to do next

- To add members, see Adding members to the admin organization
- To change the roles assigned to members see Assigning roles to members
- To delete a member, see Deleting a member

Adding members to the admin organization

As the Cloud Manager administrator, you can add and remove members of the Admin organization and assign roles to them. Once removed, a user can no longer access the Cloud Manager.

About this task

Membership in the Admin organization is required to use Cloud Manager to view and manage the topology of the API Connect cloud, to set up and manage provider organizations, and to manage other features of the cloud. There are two ways to add a member:

- Invite member - send an email invitation to a new user to register as a member of the Admin organization.
- Add member - specify the user and add them as a member of the Admin organization immediately.

One of the following roles is required to add members:

- Administrator

- Owner

Procedure

- To send an email invitation to a new user to register as a member of the Admin organization, complete the following steps:
  1. In the Cloud Manager, click Members.
     The list of current members for the Admin organization is displayed. You can view the owner of the Admin Organization by expanding View Owners.
  2. Click Add > Invite member.
  3. Enter a valid email address.
  4. Select the roles that you want to assign to the user.
  5. Click Invite. An email containing an activation link is sent to the email address inviting the person to register. The link opens the registration form. Alternately, you can copy and paste the activation link into an email; click Get Activation Link and click Copy in the pop up window.
     Note that the email server requires the Authenticate User and Authenticate Password values in order to send invitations. See Configuring an email server for notifications.

     The user is placed in Pending status, and is changed to Enabled status after they complete the registration form to activate their account. They can then log in to the Cloud Manager user interface with the API Connect user name that they specified during account activation.
     Note: If you use this option with a Local User Registry, and the user already exists in the registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. This stipulation includes users that were previously members but were deleted from the admin organization.
- To specify the user and add them as a member of the Admin organization immediately, complete the following steps:
  1. In the Cloud Manager, click Members.
     The list of current members for the Admin organization is displayed. You can view the owner of the Admin Organization by expanding View Owners.
  2. Click Add > Add member.
  3. Select the required user registry
     The remaining procedure varies according to the type of the selected user registry, as follows:
     - Local User Registry
       - Select whether the user is an Existing user or a New User.
         - For an existing user, complete the following steps:
           - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
           - Select the roles that you want to assign to the user, then click Add. The user is added and their membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.
         - For a new user, complete the following steps:
           - Enter a unique user name for the new user.
           - Supply an email address, name details, and a password.
           - Select the roles that you want to assign to the user, then click Add. The user account is created, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.
     - LDAP
       - Enter the name of a user that exists in the selected user registry.
       - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface with their LDAP user name.
     - Authentication URL and OIDC
       - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
       - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the Cloud Manager user interface.

What to do next

The new member can log in and work in Cloud Manager. The member's authorization is defined by the roles assigned to them.
For instructions on assigning roles, see Assigning roles to members. The list of roles and permissions can be viewed at User roles and permissions in the Cloud Manager UI.

Assigning roles to members

As the Cloud Manager administrator, you manage the roles that are assigned to members. A member's role determines the tasks they are authorized to perform in Cloud Manager.

About this task

Members in the Admin organization use Cloud Manager to view and manage the topology of the Cloud, to set up and manage provider organizations, and to work with other features of the API Connect Cloud. The member's role determines what tasks they are authorized to perform.
When assigning roles, you need to be aware that all members always have a basic set of permissions, which are contained in the **Member** role. The Member role is automatically granted to all members and cannot be removed. For this reason, you do not explicitly assign the Member role. You can grant additional permissions by selecting one or more roles. If additional roles are not assigned, then the user will have only the basic Member role.

One of the following roles is required to assign roles to members:

- Administrator
- Owner

Procedure

Perform the following steps to assign one or more roles to a member:

1. In the Cloud Manager, click Members.
2. The list of current members for the Admin organization is displayed.
3. To change the roles assigned to a member, select the roles by marking the check boxes next to their name. The roles will be updated instantly. The list of roles and permissions can be viewed at User roles and permissions in the Cloud Manager UI

Results

The member will be assigned the permissions for the role(s).

What to do next

The new member can work in Cloud Manager. The member's authorization is defined by the roles assigned to them.

Deleting a member

As the Cloud Manager administrator, you can delete members of the API Connect cloud administration organization. Once deleted, the user and associated roles are removed, however the user's account still remains in API Connect.

About this task

One of the following roles is required to delete a member:

- Administrator
- Owner

Note: The user account of the deleted user remains in the associated API Connect user registry. The user can still log in to the Cloud Manager user interface but only information links are available; there is no access to view or manage any resources.
The user can be added again as a member at a later time; see Adding members to the admin organization.

Procedure

To delete a member of the cloud administration organization, complete the following steps:

1. In the Cloud Manager, click Members.
2. Choose Delete from the overflow menu that is adjacent to the member you want to delete.
3. Confirm the deletion.

Results

The member is deleted and removed from the Members list in Cloud Manager.

Role Defaults overview

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

Role Defaults concepts

Role Defaults determine the base roles applied to either Provider or Consumer Organizations. The Role Defaults for the Admin organization are not configurable. Role Defaults help to standardize base roles for all organizations. Roles and permissions may be added or modified in the Role Defaults to customize the base roles as needed. Role Defaults are a "point-in-time" configuration, which effects only new organizations. You can add, delete, or modify roles (except for the Owner, Member, and Viewer roles) in the organization to customize the permissions for each organization.

Rules for editing Role Defaults:

- Edits to Role Defaults are not applied retroactively to existing organizations. Edits apply only to new organizations created after the changes are made.
- Role Defaults apply to all new organizations; roles can then be edited at the organization, catalog, and space level.
- Edits made to roles in an organization do not affect the Role Defaults. Role Defaults are the "parent" and organization roles are the "child". The child inherits from the parent, but changes do not filter up from the organization to the Role Default.

- **Managing role defaults for provider organizations**
  Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.
- **Managing role defaults for consumer organizations**
  Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

Related tasks

- Managing role defaults for provider organizations

- Managing role defaults for consumer organizations

Related information

- Creating custom roles

Managing role defaults for provider organizations

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

About this task

If you are assigned a role that has permissions to manage Cloud Settings, you can create manage the Role Defaults for the provider organizations.
One of the following roles is required to add roles to the role defaults:

- Administrator
- Topology Administrator
- Owner

Perform the following steps to add or edit roles to the Roles Defaults for provider organizations:

Procedure

1. In the Cloud Manager, click Settings.
2. Select Role Defaults.
3. Click Add to add a new role.
   a. To edit a role, select Edit from the overflow menu for the role.
   b. To delete a role from the Role Default, select Delete from overflow menu and confirm the deletion. Owner, Member, and Viewer roles cannot be deleted.
   Note: Role deletions apply to new members added after the deletion. If a current member is assigned the role, they will continue to have the permissions connected with the role. You need to manage the deletions for each member in the Members screen.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see User roles and permissions in the Cloud Manager UI.
6. Save the role.

Results

The role appears in the list of roles for the provider organization Role Defaults. It will be included in the roles that can be assigned to members for all new provider organizations. The roles can be edited for individual organizations.

Related concepts

- Role Defaults overview

Related tasks

- Managing role defaults for consumer organizations
- Assigning roles to members

Managing role defaults for consumer organizations

Role Defaults are role templates that determine the default roles created in new Provider and Consumer organizations. Cloud Manager ships with a set of pre-configured Role Defaults. As the Cloud Manager administrator, you can edit the pre-configured Role Defaults.

About this task

If you are assigned a role that has permissions to manage Cloud Settings, you can create and manage the Role Defaults for consumer organizations.
One of the following roles is required to add roles to the role defaults:

- Administrator
- Topology Administrator
- Owner

Perform the following steps to add or edit roles to the roles defaults for consumer organization:

Procedure

1. In the Cloud Manager, click Settings.

2. Select Role Defaults.
3. Click Add to add a new role.
    a. To edit a role, select Edit from the overflow menu for the role.
    b. To delete a role from the Role Default, select Delete from overflow menu and confirm the deletion. Owner, Member, and Viewer roles cannot be deleted. Note: Role deletions apply to new members added after the deletion. If a current member is assigned the role, they will continue to have the permissions connected with the role. You need to manage the deletions for each member in the Members screen.
4. Enter the Title for the role. The Name will be auto-generated.
5. Use the check boxes to select the permissions for the role. For a description of the permissions and the actions they enable, see User roles and permissions in the Cloud Manager UI.
6. Save the role.

Results

The role appears in the list of roles for the consumer organization Role Defaults. It will be included in the roles that can be assigned to members for new consumer organizations. The roles can be edited for catalogs, but permissions inherited from the Role Default cannot be deleted. Permissions can be added at the catalog level, but not deleted.

Related concepts

- Role Defaults overview

Related tasks

- Managing role defaults for provider organizations
- Assigning roles to members

Monitoring the cloud

API Connect generates events to monitor the status of your cloud.

About this task

You can monitor your API Connect cloud and configure destination targets for your analytics data by using the following tasks:

- **Accessing and configuring analytics**
  Monitor the usage and performance of API calls by using the analytics feature to view, filter, sort, and aggregate your API event data.
- **Configuring audit logging to monitor user operations**
  Configure auditing in IBM API Connect to monitor user operations such as who published which product and when.
- **Reviewing a gateway's processing status**
  Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

Accessing and configuring analytics

Monitor the usage and performance of API calls by using the analytics feature to view, filter, sort, and aggregate your API event data.

From the analytics view of the Cloud Manager UI, you can monitor the usage of your published APIs. API events from all catalogs, spaces, and organizations are captured and displayed together in the Cloud Manager analytics view. For more information about the features of the analytics view, see Viewing analytics data in Cloud Manager.

API event records

An API event is logged each time an API operation is invoked on the gateway. The API event record contains information about the API call. When an analytics service is associated with a gateway, all API invocations on that gateway are captured as API event records and sent to the analytics service. The content of the record depends on the logging policy that is set for the operation. For more information about the fields that are displayed in an API event record, see API event record fields.

Analytics configuration options

API Connect offers many configuration options to customize data collection, storage, and retention. For example, you can define filters that refine data to add or remove fields before they are stored. You can choose to offload some, or all, of the data to a third-party storage system. For more information about the analytics configuration options, see Advanced configuration options

Configuring audit logging to monitor user operations

Configure auditing in IBM® API Connect to monitor user operations such as who published which product and when.

Audit logging is useful when you want to monitor the create, read, update, delete, log-in, and log-out activity in your deployment; for example, who made changes, when, and what they changed.

Note: Actions performed by Portal administrators are not captured by the audit feature.

What events are audited?

All create, update, delete, log-in, and log-out operations that are performed by users in the Admin UI, Manager UI, Toolkit, and REST API are captured by this feature. For Portal users, all create, update, and delete operations involving applications are captured.

Beginning in API Connect 10.0.5.3, read operations can also be audited when the **audit_reads** setting is additionally enabled (using the toolkit CLI).

Attention: If you choose to enable auditing of read operations for an HTTP endpoint, be aware of that the response might return large payloads, and might contain sensitive information.
Internal calls and calls between system components are not audited because they are part of the product functionality and are not executed by users.

How are audit events logged?

Audit events are logged by sending the information about operations to one or more endpoints of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.

- HTTP endpoints:
  Audit logging messages are sent as HTTP/HTTPS requests and use the payload format defined by the Cloud Auditing Data Federation (CADF). If the endpoint requires authentication, you can configure a header that will be included in the audit logging request. Audit logging requests are issued once only, on the assumption that every endpoint is available to receive the requests. There is no error checking to ensure that requests are received and stored by the endpoints, and failed requests are not re-sent. If you suspect that audit logging requests are not successfully reaching an endpoint, you can check the server logs on the console, where any error messages are recorded.

  There are two modes that control how the auditing is sent to an endpoint: blocking and non-blocking. In non-blocking mode (the default), an audit event is sent to the endpoint and the system does not wait for a response from the endpoint. Non-blocking mode provides the best performance. Blocking mode should be used with caution: when an audit event is sent, the system pauses and waits for a response from the endpoint. Blocking mode has a negative impact on performance.

- Syslog endpoints:
  Audit logging messages are sent as syslog messages. If the endpoint requires mutual TLS, you can configure a TLS profile to send the required client certificates. Audit logging requests are issued once only, on the assumption that every endpoint is available to receive the requests. There is no error checking to ensure that requests are received and stored by the endpoints, and failed requests are not re-sent. If you suspect that audit logging requests are not successfully reaching an endpoint, you can check the server logs on the console, where any error messages are recorded.

  All syslog messages are sent as non-blocking regardless of the auditing mode, since syslog messages are designed to be fire and forget (syslog was originally a UDP protocol, which by its nature does not have any acknowledgment). Blocking is only used for HTTP endpoints.

Note: With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

What's included in the audit logging request?

- HTTP endpoints:
  The payload containing the audit logging record includes information about the user who executed the operation, the date and time of the call, the resource affected by the call, the operation performed, and the payload. If you enabled audit logging for read operations, the payload might also contain the properties from the response (based on your configuration settings).

  To see the most recent set of fields that are included in the payload, refer to the OpenAPI schema for the audit logging request in the Provider API, which you can download at https://us-south.functions.cloud.ibm.com/api/v1/web/API%20Connect%20Native_apic-on-prem/default/v10-index.http.

  HTTP example 1: sample payload for an audit logging message directed to an HTTP endpoint (IDs and users are not real):


```
{
  "id": "f6fcacb5-e8eb-4e2c-0f67-53b6a792d7f0",
  "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
  "action": "update",
  "outcome": "success",
  "eventType": "activity",
  "reason": {
    "reasonCode": "200",
    "reasonType": "HTTP"
  },
  "eventTime": "2020-02-01T15:18:16.011Z",
  "initiator": {
    "id": "/api/user-registries/38385c9f-7837-583c-01f7-9d8c37a9a80d/10de0db0-27b5-35d7-b4b5-635eabb48b4a/users/24c6cddc-
7a7e-5fc6-882d-5317d822048e",
    "name": "admin:default-idp-1/tjwatson",
    "typeURI": "service/security/account/user"
  },
  "target": {
    "id": "0beb6d21-6207-5381-b9a7-cc91a3e82c19",
    "typeURI": "tls_client_profile"
  },
  "observer": {
    "id": "target"
  },
  "requestPath": "/api/orgs/admin/tls-client-profiles/uma-tls/1.0.0",
  "requestData": {
    "url": "/api/orgs/admin/tls-client-profiles/uma-tls/1.0.0",
    "body": {
      "visibility": {
        "type": "public"
      },
      "namespace": "38385c9f-6726-472b-91f7-9d8c37a9a80d",
      "updated_at": "2020-02-01T15:18:15.924Z"
    }
  },
  "responseData": {
```

```
    "id": "0beb6d21-6207-5381-b9a7-cc91a3e82c19",
    "url": "https://my_host.example.com:3003/api/orgs/38385c9f-7837-583c-01f7-9d8c37a9a80d/tls-client-profiles/0beb6d21-
6207-5381-b9a7-cc91a3e82c19",
    "name": "uma-tls",
    "version": "1.0.0",
    "title": "Uma TLS Client Profile"
  },
  "attachments": [
    {
      "timestamp": {
        "start": "2020-02-01T15:18:15.802Z",
        "end": "2020-02-01T15:18:16.011Z"
      },
      "cloud_name": "management.example.com",
      "request_id": "f6fcacb5-e8eb-4e2c-0f67-53b6a792d7f0",
      "method": "patch",
      "operation": "tls_client_profile_updateByNameVersion",
      "summary": "Update the TLS Client Profile object by name and version",
      "resource": "tls_client_profile",
      "user": {
        "url": "/api/user-registries/38385c9f-7837-583c-01f7-9d8c37a9a80d/10de0db0-27b5-35d7-b4b5-
635eabb48b4a/users/24c6cddc-7a7e-5fc6-882d-5317d822048e",
        "context": "admin",
        "idp_name": "default-idp-1",
        "name": "tjwatson"
      },
      "registration": {
        "url": "/api/cloud/registrations/9d13c715-e3dc-55ef-92f1-1c2651c0a660",
        "type": "toolkit"
      }
    }
  ]
}
```


HTTP Example 2: sample of the `responseData` part of the message payload for a read operation (`GET /orgs`) with `list_results_limit=3`:


```
responseData:
    results_read:6
    total_results:6
    results:
        [0]:{ summary: "Administers the admin organization", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-
9ff1-79f72832d405", updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "administrator", created_at: "2023-04-
03T10:03:53.000Z", id: "e42e304e-a335-4df0-876d-682cd8a36725", type: "role", title: "Administrator", api_version: "2.0.0",
url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/e42e304e-a335-4df0-876d-682cd8a36725",
permission_urls: [<cut>] }
        [1]:{ summary: "Minimum role", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405",
updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "member", created_at: "2023-04-03T10:03:53.000Z", id:
"d415fcd9-7e22-4522-af15-c62d3094599a", type: "role", title: "Member", api_version: "2.0.0", url:
"https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/d415fcd9-7e22-4522-af15-c62d3094599a",
permission_urls: [<cut>] }
        [2]:{ summary: "Manages API provider organizations", org_url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-
9ff1-79f72832d405", updated_at: "2023-04-03T10:03:53.000Z", scope: "org", name: "organization-manager", created_at: "2023-
04-03T10:03:53.000Z", id: "4c51446d-417d-4c4a-a1be-083ecc699c2c", type: "role", title: "Organization Manager",
api_version: "2.0.0", url: "https://localhost:3003/api/orgs/18361c4c-54a2-41b2-9ff1-79f72832d405/roles/4c51446d-417d-4c4a-
a1be-083ecc699c2c", permission_urls: [<cut>] }
```


- Syslog endpoints:
  The nature of syslog messaging means that the information logged is less structured and shorter than information logged using structured HTTP endpoints. With Syslog, the information is more of a summary.

  The following sample response shows payloads for an audit logging message directed to a syslog endpoint (IDs and users are not real):


```
The user admin:default-idp-1/admin has deleted the resource Org 'alpha (Alpha title)', id 19cf81c7-7cb3-43de-b5e5-
d92cdfb26214 and url https://localhost:3003/api/orgs/19cf81c7-7cb3-43de-b5e5-d92cdfb26214
The user admin:default-idp-1/admin has deleted the resource Gateway Service 'webhook-gw-v5 (Webhook Gateway Service (v5)
title)', id 08b87e25-616d-451e-8766-bad1c7b82a42 and url https://localhost:3003/api/orgs/69c878db-4f11-4c3e-9499-
10daa25f04de/availability-zones/cb2caa3d-d619-4c2e-85f8-709956495a7b/gateway-services/08b87e25-616d-451e-8766-bad1c7b82a42
The user provider:default-idp-2/steve@example.com has created the resource Catalog 'climbon:1.0.1 (The climbon product
description)', id 665036df-28ad-447e-b8de-fe9d6967b66d and url https://localhost:3003/api/catalogs/19cf81c7-7cb3-43de-
b5e5-d92cdfb26214/8641fda7-43f8-4e59-b405-58a4844016c9/products/665036df-28ad-447e-b8de-fe9d6967b66d
The user admin:default-idp-1/admin has read the resource Cloud Setting 'cloud-setting', id settings and url
https://localhost:3003/api/cloud/settings
```


Verifying that the endpoint is receiving audit logging messages

By default, auditing operates in `non_block` mode to ensure best performance. In `non_block` mode, each audit logging message is sent immediately without waiting to ensure that the previous message was received. If you suspect that audit logging messages are not being received by the endpoint, download the server logs from the Kubernetes console and examine the apim microservices file for messages related to the connection state. Review the messages to validate that the connection was successful or to further diagnose issues.

For HTTP endpoints (not available for Syslog endpoints), you can test the connection by issuing a "test-connection" call using either the toolkit or a REST API. A response status of `204` indicates that the call was successful and the endpoint is available. The following example contains the payload from a connection test (IDs and users are not real):


```
{
  "id": "8d8eaaa2-1b46-4dcb-cadf-639d7d1f7f20",
  "typeURI": "http://schemas.dmtf.org/cloud/audit/1.0/event",
  "action": "create",
  "outcome": "success",
  "eventType": "activity",
  "reason": {
    "reasonCode": "200",
    "reasonType": "HTTP"
```

```
  },
  "eventTime": "2020-01-30T15:51:18.254Z",
  "initiator": {
    "name": "admin:default-idp-1/tjwatson",
    "typeURI": "service/security/account/user"
  },
  "target": {
    "id": "test-connection",
    "typeURI": "cloud_setting"
  },
  "observer": {
    "id": "target"
  },
  "requestPath": "/api/cloud/settings/audit-endpoint/test-connection",
  "requestData": {
    "url": "/api/cloud/settings/audit-endpoint/test-connection",
    "body": {
      "message": "hello"
    }
  },
  "responseData": {
    "name": "world"
  },
  "attachments": [
    {
      "timestamp": {
        "start": "2020-01-30T15:51:18.160Z",
        "end": "2020-01-30T15:51:18.254Z"
      },
      "cloud_name": "management.example.com",
      "request_id": "8d8eaaa2-1b46-4dcb-cadf-639d7d1f7f20",
      "method": "post",
      "operation": "cloud_setting_auditEndpointTestConnection",
      "summary": "Test connection using one of the auditing endpoints",
      "resource": "cloud_setting",
      "user": {
        "id": "/api/user-registries/dd49cebf-d3ba-5fee-92eb-f895328bc9ef/d51fa599-a5f3-50e2-a786-fafcc76daca5/users/7ff5fff8-
3d4b-3524-8687-524740580d0b",
        "context": "admin",
        "idp_name": "default-idp-1",
        "name": "tjwatson"
      },
      "registration": {
        "url": "/api/cloud/registrations/734c04f7-0fe2-3ffa-b745-81027a3b8c76",
        "type": "toolkit"
      }
    }
  ]
}
```


Tip: To ensure that audit logging messages are not lost due to an unavailable endpoint, verify each endpoint periodically.

- **Configuring the audit settings**
  Use the Cloud Manager UI or the toolkit CLI to enable auditing and configure settings.

Configuring the audit settings

Use the Cloud Manager UI or the toolkit CLI to enable auditing and configure settings.

About this task

The audit feature tracks platform-level changes in API Connect through the use of platform APIs, CLI, and Web UI. For information on tracking run-time API execution, see API Analytics.

You can configure audit settings using either the Cloud Manager UI, or the Toolkit CLI.

Using the Cloud Manager UI to configure audit settings

Use the Cloud Manager user interface to enable auditing of customer API calls.

Before you begin

Configure one or more audit endpoints where logging data can be directed, and validate that every endpoint is available by testing the connection as explained in Configuring audit logging to monitor user operations.

About this task

By default, auditing is disabled. If you enable auditing, the API Connect management node emits audit events as messages and sends them to the endpoint for storage.

Note: Auditing for read operations can only be enabled and configured using the CLI.

Procedure

1. In the Cloud Manager, click Settings > Audit Setting > Edit.
2. Click Enable to enable the audit feature, or Disable to disable it.
3. Select an audit mode:

- Non-blocking - Default value, offers the best performance. Each request is passed immediately without waiting for a response to the previous request. This is susceptible to errors if a request is dependent on the results of an earlier request whose response is not received before the new request is sent.
- Blocking - If you select this mode, expect a degradation in performance. Each request is allowed to finish execution before the next request is sent. This mode results in fewer errors when requests are dependent on the responses of earlier operations, but the delayed requests slow the network.
  Note: Blocking mode is only applied to HTTP endpoints and is ignored for Syslog endpoints.

4. Click Add and select the type of endpoint that you want to add (you must add at least one endpoint).
   Endpoints can be of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.

5. In the Endpoints section, fill in the fields to describe the new endpoint:

   HTTP
   - URL - Required, must not be empty. The URL of the server where audit logging messages will be sent.
     If you previously saved the audit settings with an empty URL and experienced problems, you can edit the URL to correct it.
   - TLS client profile - Optional. The URL of the server where the endpoint's TLS client profile is stored.
   - Header name and Header value - Optional. Multiple headers are allowed (click Add to add another header). For each header, specify the header name and the header value.

   Syslog UDP
   - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com.
     Do not include the **`syslog://`** protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
   - Port - Optional. The port number where audit logging messages should be directed.

   Syslog TCP
   - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com.
     Do not include the **`syslog://`** protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
   - Port - Optional. The port number where audit logging messages should be directed.

   Syslog TLS
   - Host name - Required, must not be empty. The host name of the server where audit logging messages will be sent; for example: www.example.com.
     Do not include the **`syslog://`** protocol with the host name. When audit logging messages are directed to the host server, the protocol is added to the host name programmatically.
   - Port - Optional. The port number where audit logging messages should be directed.
   - TLS client profile - Optional. The URL of the server where the endpoint's TLS client profile is stored.

6. Verify that you can reach the endpoint by clicking Test.
7. To specify another endpoint, click Add at the beginning of the "Endpoints" section.
8. When you finish adding endpoints, click Save to save your settings.

Results

With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

What to do next

If you suspect that audit logging messages are not being received by the endpoint, see Verifying that the endpoint is receiving audit logging messages for information on validating the connection.

Using the CLI to configure audit settings

Use the IBM API Connect toolkit CLI to enable auditing of API calls.

Before you begin

Configure one or more audit endpoints where logging data can be directed, and validate that every endpoint is available by testing the connection as explained in Configuring audit logging to monitor user operations.

About this task

By default, auditing is disabled. If you enable auditing, the API Connect management node emits audit events as messages and sends them to the endpoint for storage.

Use the following **`audit_setting`** options to enable auditing with the CLI toolkit:

- **`enable`** - Boolean, defaults to **`false`** (no audit). Set to **`true`** if you want to enable auditing.
- **`mode`** - Enumerated, defaults to **`non_block`**. The audit mode determines how audit requests are timed when they are sent to the endpoint:
  - **`non_block`** - Default value, offers the best performance. Each request is passed immediately without waiting for a response to the previous request. This is susceptible to errors if a request is dependent on the results of an earlier request whose response is not received before the new request is sent.
  - **`block`** - If you select this mode, expect a degradation in performance. Each request is allowed to finish execution before the next request is sent. This mode results in fewer errors when requests are dependent on the responses of earlier operations, but the delayed requests slow the network.
    Note: Block mode is only applied to HTTP endpoints and is ignored for Syslog endpoints.
- **`audit_reads`** - Use this setting to enable auditing of read operations:
  - **`enable`** - Boolean, defaults to **`false`** and no audit events are sent to the endpoint for read operations. Set to **`true`** if you want to enable audit logging for read operations.
  - **`response_data_to_http_endpoints`** - Configure additional settings for HTTP endpoints only (the settings do not apply to Syslog endpoints).
    - **`enable`** - Boolean, defaults to **`false`** and audit events for GET operations are sent to the HTTP endpoint with no response data in the event. Set to **`true`** if you want to include the response data.
      Attention: If you choose to include the response data, be aware that the response might return large payloads, and might contain sensitive information.

- **`list_results_limit`** - Integer, must be 0 or larger. Use this setting to specify the maximum number of results of **`GET list`** operations to include in the audit message (default is 0 results).

  If **`audit_reads`** is enabled, then the audit message for a **`GET`** of *a single object* (for example, **`GET /orgs/admin`**) always includes all of the properties of the response object.

  If the **`list_results_limit`** is set to 0, then the audit message for a **`GET`** to *list objects* (for example, **`GET /orgs`**) does not include any objects from the GET response. If you set the **`list_results_limit`** to a value greater than 0, then the audit message includes all properties from the first **`list_results_limit`** number of objects from the **`GET /orgs`** response.

- **`endpoints`** - Array of strings. You must specify at least one endpoint where the audit logging messages are sent. Endpoints can be of the following types: HTTP, Syslog UDP, Syslog TCP, or Syslog TLS.
  For HTTP and Syslog TLS endpoints, you can optionally include a TLS client profile URL. For HTTP endpoints, you can additionally include one or more optional headers. For example, if the endpoint requires an authorization header, you can include it in the endpoint setting.

  - **`endpoint_type`** - String. Required. Allowed values are: **`http`**, **`syslog_udp`**, **`syslog_tcp`** or **`syslog_tls`**.
  - **`endpoint`** - String. Required, must not be empty or blank. The URL of the server where audit logging messages will be sent. Include the protocol (**`http://`** or **`syslog://`**) and the port, if applicable. For example, **`syslog://<hostname>:<port>`**.
  - **`tls_client_profile_url`** - String. Optional; supported only for HTTP and Syslog TLS endpoints. The URL of the server where the endpoint's TLS client profile is stored.
  - **`headers`** - String. Optional; supported only for HTTP endpoints. Multiple headers are allowed. For each header, specify the header name and the header value using the format shown in the example.

The endpoint setting can be lengthy, so uploading the settings in a file helps to avoid errors. Format the settings as a .yaml file (you can choose the file name) and upload the file using the toolkit CLI. The following example enables auditing in non-blocking mode with multiple endpoints to demonstrate how the settings should be formatted.

```
audit_setting:
  enabled: true
  mode: non_block
  audit_reads:
    enabled: true
    response_data_to_http_endpoints:
      enabled: true
      list_results_limit: 0
  endpoints:
    - endpoint_type: syslog_udp
      endpoint:
        endpoint: syslog://syslog-audit-endpoint.example_1.com:514
    - endpoint_type: syslog_tls
      endpoint:
        endpoint: syslog://syslog-tls-audit-endpoint.example_1.com:1601
        tls_client_profile_url: https://security.tls_example.com/api/orgs/d8eac736-b4e2-3aa6-c7ff-978a4cd19d0b/tls-client-
profiles/19d0f087-b4e2-3aa6-926b-9f6d51d028c3
    - endpoint_type: http
      endpoint:
        endpoint: https://mutt-audit-endpoint.example_1.com/auditing
    - endpoint_type: http
      endpoint:
        endpoint: https://auditing-endpoint.example_1.com/inputs/64e38c02-acd9-6530-76f8-dd0f336d7b11
        tls_client_profile_url: https://security.tls_example.com/api/orgs/d8eac736-b4e2-3aa6-c7ff-978a4cd19d0b/tls-client-
profiles/19d0f087-b4e2-3aa6-926b-9f6d51d028c3
      headers:
        authorization: Basic c3Bvb246c2ltb25rYXRpZXVtYQ==
    - endpoint_type: http
      endpoint:
        endpoint: https://auditing-endpoint.example_2.com/inputs/64e38c02-acd9-6530-76f8-dd0f336d7b11
      headers:
        authorization: Basic c3Bvb246c2ltb24=
        x-header: ForYourEyeOnly
```

Procedure

1. Create a YAML file with the audit settings as shown in the example.
   For information on using input files with the toolkit CLI, see Reading input from the command line.

2. Run the following command to log in to a management server as a member of the cloud administration organization:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm admin/identity_provider
   ```

   Cloud settings are stored on a management server and you must log in to the server before you can access the settings with the CLI.

   You can determine which identity provider to use in the **`--realm`** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The **`title`** value should enable you to determine which identity provider to use; you can then copy the corresponding **`--realm`** parameter directly from the displayed **`realm`** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at

creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For more information on logging in, see Logging in to a management server.

3. Upload the YAML file with the auditing settings by running the following command:

   ```
   apic cloud-settings:update --server mgmt_endpoint_url my_audit_settings.yaml
   ```

4. Run the following command to log out of the management server:

   ```
   apic logout --server mgmt_endpoint_url
   ```

Results

With the microservices nature of API Connect, any changes to the audit settings can take up to five minutes to be fully honored by all of the microservices supporting it.

What to do next

If you suspect that audit logging messages are not being received by the endpoint, see Verifying that the endpoint is receiving audit logging messages for information on validating the connection.

Reviewing a gateway's processing status

Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

About this task

Sometimes you might want to check the status of an API event that is being processed by the gateway. For example, you might want to ensure that a Product was published before and is available on the gateway before you invoke one of its APIs. Or perhaps it appears that API events are being processed too slowly and you want to see if all of the events for a particular gateway service are affected.

You can review a gateway's processing status in the API Manager user interface, or by using the toolkit CLI to retrieve the information.

Notes:

- You can only obtain processing status information for the DataPower® API Gateway. This feature is not supported for the DataPower Gateway (v5 compatible) service.
- Processing information is collected for each Catalog. If you view information for a particular Space, the display shows the results for the entire Catalog instead.

You can view the gateway processing status using the UI or the CLI.

Using the UI to view the gateway processing status

About this task

Use the API Manager user interface to view the processing status for all of the gateway services associated with a particular Catalog or Space.

Procedure

1. Open the Gateway services page:
   a. Log in to API Manager.
   b. In the navigation pane, click Manage.
   c. Select a Catalog.
   d. Click the Catalog settings tab.
   e. On the Settings page, click Gateway services.
   A list of all of the gateway services for the selected Catalog, including all of its Spaces, displays the gateway type and its endpoint URL. In addition, a status icon displays for each service name to provide a quick indication of that service's current status:

   - Normal: Between 0 and 10 events are waiting to be processed
   - Increased: Between 10 and 20 events are waiting to be processed
   - High: More than 20 events are waiting to be processed

   You can view details about a service's status to gain insight into possible issues.

2. On the Gateway services page, click ⋮ next to a service and select View status.
   The Gateway processing status page displays a list of the events that were sent to the gateway. For each event, you can see the following information:

   - Event type: The action performed by the event; for example, subscribing to a product, deleting an application, or creating a new consumer organization.
   - Category: The feature area affected by the event; for example, subscriptions, applications, or consumer organizations.
   - Time: The time when the event was processed by the gateway service.
   - Status: The current event's processing status; for example, Queued (not yet sent for processing), Sent (but not processed yet), or Processed (complete). The number of events in the Queued and Sent states determine the status level for the service.

   The display includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

Using the CLI to obtain a gateway's processing status

About this task

You can use the developer toolkit CLI to obtain information about the processing status for the Gateway services that are enabled in your Catalogs and Spaces.

The processing status information is returned in the following fields:

Field	Description
`service_up_to_date`	If `true` then all outstanding events have been processed by the Gateway service.
`service_state`	An overall status summary of the various states of events and processing by the Gateway service: • `green` indicates normal status with between 0 and 10 events waiting to be processed • `orange` indicates that between 10 and 20 events are waiting to be processed • `red` indicates that more than 20 events are waiting to be processed
`last_processed_event`	Details of the last event that was processed by the Gateway service, helping to provide a timeline of when the last event was processed, and some statistics about how long ago it was and how long the processing took.
`last_sent_event`	Details of the last event that was sent to the gateway service
`number_of_outstanding_sent_events`	The number of events that have been sent but not yet processed by the Gateway service.
`number_of_outstanding_queued_events`	The number of events that have been queued but not yet sent to the Gateway service.

The following example shows a "green" response that includes the status information and the list of sent events:

```
gateway_processing_status:
  service_up_to_date: false
  service_state: green
  last_processed_event:
    event_id: 40e1a564-2aea-429b-9d35-f588218688a1
    title: None provided
    processed_at: '2020-06-02 22:40:04.671+00'
    generated_at: '2020-06-02 22:39:33.856769+00'
    filter: product_replace
    event_processing_time: 24526
    elapsed_time_since_last_processed_event: 13682
  last_sent_event:
    event_id: 3bb52860-29f8-41a9-93d0-c9465a0a30e0
    title: None provided
    sent_at: '2020-06-02T22:40:15.820Z'
    generated_at: '2020-06-02 22:40:12.980079+00'
  number_of_outstanding_sent_events: 9
  number_of_outstanding_queued_events: 0
events:
  sent_events:
    - title: >-
        Update Product (name-version:lts-cat0-product-14:1.0.0, id:
        0df5bf4b-6706-4c36-aa0c-9c8ba2f15254) by user shavon0
      filter: product_update
      generated_at: '2020-06-02 22:39:45.430744+00'
    - title: >-
        Update Product (name-version:lts-cat0-product-64:1.0.0, id:
        4795106e-787d-41e6-bb94-9e6506719529) by user shavon0
      filter: product_lifecycle
      generated_at: '2020-06-02 22:39:46.11499+00'
    - title: None provided
      filter: product_replace
      generated_at: '2020-06-02 22:39:46.862433+00'
    - title: >-
        Update Product (name-version:lts-cat0-product-15:1.0.0, id:
        6c7bf5ef-dca4-4b63-8060-2cc3c4d91d3b) by user shavon0
      filter: product_update
      generated_at: '2020-06-02 22:39:58.397278+00'
    - title: >-
        Update Product (name-version:lts-cat0-product-65:1.0.0, id:
        4019df05-27a6-4ac2-83c8-6dd5da100105) by user shavon0
      filter: product_lifecycle
      generated_at: '2020-06-02 22:39:59.02833+00'
    - title: None provided
      filter: product_replace
      generated_at: '2020-06-02 22:39:59.740152+00'
    - title: >-
        Update Product (name-version:lts-cat0-product-16:1.0.0, id:
        4435a953-3c8d-4814-ada8-c71409cabcaf) by user shavon0
      filter: product_update
      generated_at: '2020-06-02 22:40:11.571056+00'
    - title: >-
        Update Product (name-version:lts-cat0-product-66:1.0.0, id:
        fb9239d5-6064-4500-9a7b-63b841f5321a) by user shavon0
      filter: product_lifecycle
      generated_at: '2020-06-02 22:40:12.199838+00'
    - title: None provided
      filter: product_replace
      generated_at: '2020-06-02 22:40:12.980079+00'
  queued_events: []
```

The following example shows a "red" response:

```
gateway_processing_status:
  service_up_to_date: false
  service_state: red
  last_processed_event:
    event_id: 07387d98-dbfc-494b-af12-d7db91208e4c
    title: >-
      Update Product (name-version:lts-cat0-product-49:1.0.0, id:
```

```
    25eba540-0cb1-4808-be3d-f22f65540cb2) by user shavon0
  processed_at: '2020-06-02 22:32:10.838+00'
  generated_at: '2020-06-02 22:31:43.38615+00'
  filter: product_del
  event_processing_time: 22597
  elapsed_time_since_last_processed_event: 18427
last_sent_event:
  event_id: 93d0bf8f-ebfb-4a8b-aa22-7f805719d5c8
  title: >-
    Update Product (name-version:lts-cat0-product-11:1.0.0, id:
    d0bf799f-7f3a-4546-b267-a85c6a7f4b10) by user shavon0
  sent_at: '2020-06-02 22:32:28.426+00'
  generated_at: '2020-06-02 22:32:25.505109+00'
number_of_outstanding_sent_events: 37
number_of_outstanding_queued_events: 1
```

The result includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

Procedure

To obtain processing status information for a Gateway service, complete the following steps:

1. Log in to your API Connect Management server as a provider organization member, by using the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   where:
   - *mgmt_endpoint_url* is the platform API endpoint URL.
   - *user_ID* is the user ID of your provider organization account, and is the same as the user ID that you use to log in to the API Manager user interface.
   - *password* is the password for your administrator account.
   - *identity_provider* is the identity provider that is used to authenticate API Manager users.

   For example:

   ```
   apic login --server platform-api.myserver.com --username myuser --password password --realm provider/myldap
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details on how to log in to your management server from the CLI, see Logging in to the management server.

2. Obtain the Gateway processing status for a Gateway service that is enabled in a Catalog by running the following command:

   ```
   apic configured-gateway-services:get --server mgmt_endpoint_url --scope catalog --org organization_name --catalog
   catalog_name gateway_service_name --fields "add(gateway_processing_status)" --output -
   ```

   where:

   - *mgmt_endpoint_url* is the platform API endpoint URL.
   - The `--scope scope` parameter specifies `catalog` as the scope of the request.
   - *organization_name* is the name of the provider organization that contains the Catalog.
   - *catalog_name* is the name of the Catalog in which the Gateway service is enabled.
   - *gateway_service_name* is the name of the Gateway service whose processing status you want to obtain.
   - The `--fields "add(arguments)"` parameter can include either, or both, of the following arguments (separated with a comma):
     - `gateway_processing_status`: returns the total number of sent events, queued events, and processed events
     - `events`: returns lists of sent events with details about each event

   For example:

   ```
   apic configured-gateway-services:get --server platform-api.myserver.com --scope catalog --org myorg --catalog sandbox my-
   gateway-service --fields "add(gateway_processing_status)" --output -
   ```

Changing your Cloud Manager password and profile information

You can change your Cloud Manager password and update your profile information.

Procedure

To change your password or profile information, follow these steps:

1. Log in to the Cloud Manager user interface.

2. Click the User icon , and then select My Account.
3. In the Profile section, enter a new email address and First Name and Last Name for the account owner.
4. Click Save.
5. In the Change password section, enter your current password and your new password, and confirm the new password.
   The password must contain six characters at a minimum. It must contain at least three of the following types of characters:
   - Lowercase letters
   - Uppercase letters
   - Numbers
   - Characters from the following list of allowed punctuation and special characters:

     ```
     !
     \"
     #
     $
     %
     &
     \
     (
     )
     *
     +
     ,(comma)
     - (dash/hyphen)
     .(period)
     /
     :(colon)
     ;(semi-colon)
     <
     =
     >
     ?
     @
     [
     \\
     ]
     ^
     _ (underscore)
     ` (back quote/grave accent)
     {
     | (pipe)
     }
     ~ (tilde)
     ```

   The password cannot contain more than two consecutive repeating characters.
6. Click Save.

Results

Your profile information and password are changed.

Related concepts

- Configuring and managing your server environment

Resolving login problems by increasing HTTP header size

You can resolve login problems for the Cloud Manager UI by increasing the maximum HTTP client header size.

About this task

Login attempts to the user interface might fail with error **`502 (Bad Gateway)`**. This error can occur when logging in from a browser that has a large number of cookies. The error occurs because size of the login request exceeds the maximum HTTP client header size.

The maximum HTTP client header size is limited for security reasons. To workaround this issue, you can clear the browser cache and cookies, or open an incognito window from the browser, and then retry the login.

Alternatively, you can increase the maximum HTTP client header size. Use caution when increasing the maximum size because larger headers can raise a security risk to your system.

Procedure

1. **`SSH`** to your management system or appliance.
2. Enter the following commands:

   ```
   kubectl get deployment -n namespace  | grep apim
   kubectl edit deployment -n namespace management-apim
   ```

3. In the **`env`** section of the deployment configuration, change **`--max-http-header-size=12000`** to a larger value that works for your environment. For example:

   ```
   - name: NODE_OPTIONS
           value: --max-old-space-size=8094 --max-http-header-size=16000
   ```

4. Save the updated configuration.
5. Run `kubectl get pod -n namespace` a few times until the new `management-apim` pods are up and running.

Onboarding a new admin for Cloud Pak for Integration

Use the API Connect toolkit CLI to create a new admin account for use with Cloud Pak for Integration.

About this task

If you changed the local admin account for Cloud Pak for Integration, you must create a new admin user in API Connect to ensure that the new Cloud Pak for Integration admin can log in to API Connect with the Common Services User Registry.

Procedure

1. Download and install the API Connect toolkit CLI and credentials file as explained in Installing the toolkit.
2. Log in to the toolkit CLI.
   Logging in to the toolkit CLI requires the endpoint URL for your API Connect management server. In Cloud Pak for Integration, the URL indicates the location of the management server within the Cloud Pak for Integration deployment.

   a. Determine the URL to the API Connect management server by running the following command:

   ```
   oc -n APIC_namespace get mgmt APIC_instance -o jsonpath="{.status.zenRoute}" && echo ""
   ```

   The response looks like the following example URL:

   ```
   apic-mgmt-admin-apic.deve-cip-hlag-8e86d3798137c73f524cf9-0000.eu-de.containers.appdomain.cloud
   ```

   b. Log in to the API Connect toolkit CLI by running the following command:

   ```
   apic -s API_Connect_URL login -u admin -p <password> -r admin/default-idp-1
   ```

   where ***API_Connect_URL*** is the URL of the management server in your API Connect instance.

3. Create a new admin account by completing the following steps.
   Use the toolkit CLI to create an admin account in the Common Services User Registry and then add the new admin to the API Connect **administrator** role.

   a. Create a YAML file called newCp4iAdmin.yaml with the following information for the new admin account.

   ```
   username: <new_cp4i_admin_username>
   first_name: CP4I
   last_name: Administrator
   ```

   b. Create the CP4i admin account by running the following command:

   ```
   apic -s API_Connect_URL users:create -o admin --user-registry common-services newCp4iAdmin.yaml
   ```

   c. Verify that the account was created by running the following command:

   ```
   apic -s API_Connect_URL users:get -o admin --user-registry common-services <new_cp4i_admin_username> --output -
   ```

   d. Get the new admin account's URL by running the following command:

   ```
   apic -s API_Connect_URL users:list -o admin --user-registry common-services | grep <new_cp4i_admin_username> | awk
   '{print $4}'
   ```

   e. Get the URL of the API Connect **administrator** role by running the following command:

   ```
   apic -s API_Connect_URL roles:list --scope org -o admin | grep -w administrator | grep -v topology | awk '{print $2}'
   ```

   f. Create a YAML file called newcp4iadminmember.yaml with the following content:

   ```
   name: <new_cp4i_admin_username>
   user:
     url: <user_URL>
   role_urls:
     - <role_URL>
   ```

   g. Use the newcp4iadminmember.yaml file to add the new admin as a member of the API Connect instance's admin organization.
   Create the new member by running the following command:

   ```
   apic -s API_Connect_URL members:create --scope org -o admin newcp4iadminmember.yaml
   ```

   h. Verify that the new admin account works by using it to log in to the Cloud Manager interface in your API Connect instance.

4. Remove the old Cloud Pak for Integration admin user from API Connect.
   After you verify that the new Cloud Pak for Integration admin can log into the API Connect Cloud Manager interface, delete the previous Cloud Pak for Integration admin account.

   a. Remove the previous admin from the membership of the API Connect admin organization by running the following command:

   ```
   apic -s API_Connect_URL members:delete --scope org -o admin cs-admin
   ```

   When you initially deployed API Connect as a component of Cloud Pak for Integration, the admin username was **admin** and the member in the API Connect admin organization was **cs-admin**. If the **cs-admin** member does not exist in the admin organization, then the Cloud Pak for Integration admin user was changed after deployment. In this case, the member uses the same name as admin account's user name. Delete the member with that name:

```
apic -s API_Connect_URL members:delete --scope org -o admin <old_cp4i_admin_username>
```

b. Delete the previous Cloud Pack for Integration admin user account from the Common Services User Registry by running the following command:

```
apic -s API_Connect_URL users:delete -o admin --user-registry common-services <old_cp4i_admin_username>
```

Extending the Gateway server behavior

To support your enterprise requirements, you can extend the Gateway servers within IBM® API Connect to provide extra enforcement behavior.

Before you begin

Before you develop your extensions, consider the guidelines in the following topics depending on which type of gateway you are using:

- Gateway extension guidelines - DataPower Gateway (v5 compatible)
- Gateway extension guidelines - DataPower API Gateway

For information on the different types of gateway, see API Connect gateway types.

About this task

API Connect Gateway servers uses a subset of DataPower® enforcement capabilities. You can supply DataPower Extensions to customize these enforcement capabilities further.

The DataPower extensibility function can be used to perform actions that include schema validation, antivirus scanning, message filtering, authentication, and authorization, token translation, message enrichment, encryption & decryption, digital signing, and validation and message transformation. For more information, see IBM DataPower documentation.

Important: Gateway server extensions must not be used to modify objects that cause the `apic-gw-service` object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the `apic-gw-service` object to restart, lead to unpredictable behavior in the Gateway service.

Procedure

To extend the default enforcement capabilities that are provided in API Connect for the Gateway server, complete the following steps.

1. In your IBM DataPower environment, develop the configuration that you want to add to your Gateway server.
2. Test your enforcement configuration before you add the configuration to the Gateway server.
3. When you complete the IBM DataPower configuration, save the enforcement objects and files as a DataPower exported .zip file.
   The package file is ready to be uploaded to the Gateway server.
4. Copy your exported configuration .zip file to a centralized file system ready for upload to the Gateway server.

What to do next

Upload your extension to the Gateway server; see Configuring your Gateway server extensions.

- **Gateway extension guidelines - DataPower Gateway (v5 compatible)**
  Before you develop your extensions to the DataPower Gateway (v5 compatible), consider these guidelines.
- **Gateway extension guidelines - DataPower API Gateway**
  Before you develop your extensions to the DataPower API Gateway, consider these guidelines.
- **Configuring your Gateway server extensions**
  You can add extra IBM DataPower enforcement capabilities to a Gateway service by uploading a .zip file that defines the required extension behavior, and then enabling the extension in IBM DataPower.

DataPower Gateway (v5 compatible)

Gateway extension guidelines - DataPower Gateway (v5 compatible)

Before you develop your extensions to the DataPower® Gateway (v5 compatible), consider these guidelines.

Note: These guidelines apply to the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Gateway extension guidelines - DataPower API Gateway.
For information on the different types of gateway, see API Connect gateway types.

You can download sample gateway extensions from https://github.com/ibm-apiconnect/dp-extensions.

You can create DataPower Processing Rules that can extend the enforcement behavior of the IBM® API Connect Gateway server at the following locations:

- `pre-request` extension:
  - Before the Gateway server begins to use the policies in the assembly to process the request.
- `post-request` extension:
  - After the Gateway server processes all of the policies in the assembly up to the proxy policy, if a proxy policy is used.
  - After the Gateway server processes all policies in the assembly, but before any catch logic is processed, if a proxy policy is not used.
- `post-response` extension:
  - After the Gateway server processes all of the remaining policies in the assembly after the proxy policy (including catch logic), but before the response is returned to the client application, if a proxy policy is used.

    - After the Gateway server processes the catch logic, but before the response is returned to the client application, if a proxy policy is not used.
- **post-error** extension:
    - If an error occurs, then before the Gateway server returns the error response to the client application.

To configure the Gateway server to call your extension Processing Rule, you must create an XML file that indicates the extension location and Processing Rule name. For example,

```
<extensions>
<extension location="pre-request">CustomRule1</extension>
<extension location="post-request">CustomRule2</extension>
<extension location="post-response">CustomRule3</extension>
<extension location="post-error">CustomRule4</extension>
</extensions>
```

The **<extension>** element entries are optional for any of the locations. Refer to the Gateway server Extension schema for the XML file syntax.

An extension Processing Rule can be applied to a specific organization or to all of the organizations and Catalogs. If the **<extension>** element does not have a tenant attribute, then the **<extension>** element is applied to all of the organizations. The following example shows an **<extension>** element without a tenant attribute.

```
<extension location="post-error">gen_error_handling</extension>
```

To apply an extension Processing Rule to a specific organization, enter the organization name as the value for the tenant attribute in the **<extension>** element. The following example shows an <extension> element with a tenant attribute value of *organization1*.

```
<extension location="post-error" tenant="organization1">organization1_error_handling</extension>
```

If you want to exclude a specific organization in an extension Processing Rule, enter the organization name as the value for the tenant attribute in an empty **<extension>** element. Add the empty **<extension>** element immediately after the custom rule **<extension>** element that you want to exclude the organization from. The following example shows an empty <extension> element with a tenant attribute value of *organization1*.

```
<extension location="pre-request">a_specified_CustomRule</extension>
<extension location="pre-request" tenant="organization1"></extension>
```

Important: An extension Processing Rule with a specified tenant attribute takes precedence over an **<extension>** element that applies to all of the organizations. Also, only one extension Processing Rule is applied to each extension location.
This XML file must be saved to the following location and included in your DataPower exported configuration .zip file:

```
local:///ext/extensions.xml
```

There can be only one DataPower exported configuration .zip file added to a Gateway server in API Connect.

CAUTION:

- Do not modify these files directly in the DataPower API Gateway file system, use the gateway extension mechanism described on this page.
- Do not apply gateway extension modifications to configuration in the local:/config-sequence.cfg file.
- Do not use Gateway server extensions to modify objects that cause the **apic-gw-service** object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the **apic-gw-service** object to restart, lead to unpredictable behavior in the Gateway service.

DataPower configuration restrictions

All Processing Rules have read access to the INPUT context.

Processing Rules must not rely on context variables that are created by the IBM Gateway server enforcement configuration, because those configuration variables might change in the future.

All Processing Rules except Before Request can transform or alter the message flowing through the Gateway server. Ensure that the Processing Rule returns the desired message output context back to the Gateway server at the end of processing.

To avoid name conflicts, all DataPower configuration object names prefixed with **webapi** are reserved for IBM use.

The following folders cannot be modified:

- **local:///isp/***
- **local:///gwapi/***

As a best practice, avoid adding asynchronous actions in your custom Processing Rules because they increase the use of memory per transaction.

Gateway extension guidelines - DataPower API Gateway

Before you develop your extensions to the DataPower® API Gateway, consider these guidelines.

Note: These guidelines apply to the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Gateway extension guidelines - DataPower Gateway (v5 compatible).
For information on the different types of gateway, see API Connect gateway types.

Introduction

You apply an extension to a DataPower API Gateway by uploading one or more .cfg files that define the required extension behavior. These .cfg files contain DataPower API Gateway CLI commands. For details of the full set of CLI commands that are available, see the DataPower API Gateway CLI command documentation.

You package the .cfg files in a .zip file, together with any additional required files that are referenced from CLI commands; for example, a .json file that contains the OpenAPI definition for an API. The .cfg files are processed in alphanumeric order by file name.
Note:

- All the .cfg files must be at the root of the .zip file. If you are attempting to use an API Connect Version 2018 gateway extension that has files in sub-directories, see Gateway extensions manifest for information on how to deploy multiple extensions in a single package.
- On Mac OS X, run the **zip** command on the command line with the following flags:

  ```
  -x ".DS_Store" -x "__MACOSX"
  ```

  Including the flags prevents the auto-generated .DS_Store and __MACOSX files from being added to the zip.

You upload the extension .zip file to an API Connect Gateway service, and enable the extension, as described in Configuring your Gateway server extensions. Gateway server extensions can be deployed without having to restart the DataPower API Gateway.

To control how gateway extensions are applied to the DataPower API Gateway, you can include a Gateway extensions manifest as a manifest.json file. The manifest allows you to deploy multiple extensions of different types in a single package. For more information, see Gateway extensions manifest.

Using CLI commands in a Gateway extension

Although you can use any DataPower API Gateway CLI commands in your extension .cfg files, typically you use commands that modify API Connect objects that have previously been deployed to the DataPower API Gateway.

When API Connect objects are deployed to a DataPower API Gateway, .cfg are created on the Gateway that define the configuration of those objects. By examining those .cfg on the Gateway, you can determine the precise names of API Connect objects that you want to modify, and the current values of the properties of those objects. You can access these .cfg files by completing the following steps:

1. Log in to the DataPower administrative user interface; for the Graphical Interface, select WebGUI rather than Blueprint Console.
2. Switch to the DataPower application domain for API Connect if necessary.
3. Under Files and Administration, click File Management.
4. Expand the temporary: folder to locate the .cfg files. The files for API Connect objects are named according to their containing provider organization and Catalog, as follows:

   20.*provider-org-name_catalog-name*_collection.cfg

   The prefix **20** controls the position of the file in the overall processing sequence, which is in alphanumeric order by file name.
   Restriction:
   - Do not modify these files directly in the DataPower API Gateway file system. Instead, use the gateway extension mechanism described on this page.
   - Do not apply gateway extension modifications to configuration in the local:/config-sequence.cfg file.
   - Do not use Gateway server extensions to modify objects that cause the **apic-gw-service** object to restart, such as TLS Profiles, and Gateway Peering. Extensions that cause the **apic-gw-service** object to restart, lead to unpredictable behavior in the Gateway service.

Note: Modifying a .cfg file triggers the execution of all other .cfg files that run commands on the same object.

Config-Sequence tracks all of the objects that it manages through the .cfg files. Any change to a .cfg file that causes config-sequence to run commands on an object automatically executes all other .cfg files that run commands on the same object.

Example: If a new product is published to catalog A, then the .cfg file for that collection in temporary://config will be modified by the apic-gw-service to reflect the new product. Because that .cfg file was changed, config-sequence will execute every other .cfg file that also runs operations on catalog A, in the sequence of locations defined in the config-sequence object. If there are 2 .cfg files in the same location that run commands on catalog A, they run in alphabetical order.

Example 1 - change the scope of the rate limit for a Plan

Suppose the DataPower API Gateway configuration for a Plan has the following CLI command:

```
api-plan myorg_sandbox_financial-services_1.0.0_basic
```

The **api-plan** command has the following syntax:

api-plan *plan_name*

and creates a Plan of the specified name, or modifies a Plan of the same name if it already exists. By using the specified name, you can add **api-plan** commands in your Gateway extension .cfg files to modify the Plan.
Note: The names of API Connect objects in the DataPower API Gateway configuration are derived from their configuration in API Connect. For example, a Plan name has the following structure:

provider-org-name_catalog-name_product-name_product-version_plan-name

where:

- *provider-org-name* is the name of the provider organization.
- *catalog-name* is the name of the Catalog in that provider organization.
- *product-name* is the name of the Product that contains the Plan.
- *product-version* is the version of the Product.
- *plan-name* is the name of the Plan.

The default scope to which a Plan rate limit applies is per application. To change this setting on the Plan in this example, so that the rate limit scope is per client ID, add the following command to a .cfg in your Gateway extension:

```
api-plan myorg_sandbox_financial-services_1.0.0_basic
  rate-limit-scope per-client-id
exit
```

For more information on configuring Plans by using DataPower API Gateway CLI commands, see API Plan commands.

Example 2 - add a path to an API definition

The .cfg file in this example performs the following actions:

1. Uses the **top** and **configure terminal** commands to reset CLI processing.
2. Uses the **api-operation** command as follows:
   a. Creates a new operation called `bank_branches_get_operation`.
   b. Uses the **reset** command to set all properties to their default values.
   c. Sets the operation method to `GET`.
3. Uses the **api-path** command as follows:
   a. Creates a new path called `bank_branches_path`.
   b. Uses the **reset** command to set all properties to their default values.
   c. Sets the path URL segment to `/details`.
   d. Adds the previously created operation `bank_branches_get_operation` to the path.
4. Uses the **api-definition** command as follows:
   a. Modifies the existing API definition called `myorg_sandbox_myapi_1.0.0`.
   Note: The name of the API definition was derived from its configuration in API Connect when it was published to the DataPower API Gateway, and has the following structure:

   provider-org-name_catalog-name_api-name_api-version

   where:
   - *provider-org-name* is the name of the provider organization.
   - *catalog-name* is the name of the Catalog in that provider organization.
   - *api-name* is the name of the API definition in API Connect.
   - *api-version* is the version of the API.

   b. Adds the previously created path `bank_branches_path` to the API definition.
   Note: The **reset** command is not used here because that would reset all the current settings on the `myorg_sandbox_myapi_1.0.0` API definition, whereas the requirement is to add a new path to the existing configuration of the API definition.

The .cfg file is as follows:

```
top; configure terminal

api-operation bank_branches_get_operation
  reset
  method GET
exit

api-path bank_branches_path
  reset
  path "/details"
  operation bank_branches_get_operation
exit

api-definition myorg_sandbox_myapi_1.0.0
  path bank_branches_path
exit
```

For more information on the DataPower API Gateway CLI commands used in this example, see the following pages:

- Initial login and common commands
- API Operation commands
- API Path commands
- API Definition commands

Gateway extension limitations

This section details the limitations of the extensions to the DataPower API Gateway.

- If you use an extension to create a user-defined policy and then delete the extension, the extension is removed from the filesystem, but the configuration sequence is not triggered to clean up related files. For the policy to be cleaned up, you must send a modified extension that explicitly removes the user-defined policy from `apic-gw-service`, as in the following example.

  ```
  apic-gw-service
      no user-defined-policies udp-basic_1.0.0
  exit
  ```

- `user-defined-policy-yaml` policies that use or reference another policy can be deployed only after the referenced policy has been advertised to API Manager. The deploy type must be set to `deferred` and the API Connect Gateway Service must be restarted.
- Deleting an extension that is used to create a user-defined policy using a .cfg file does not remove the policy reference in the `apic-gw-service` object. The policy is not removed from the assembly palette until a reboot or restart of the Gateway pod. To properly remove the policy, send an `apic extensions:update` command that explicitly removes the policy from the `apic-gw-service` prior to sending the `apic extensions:delete` command.
- v5 policies that contain files outside of the `/policy` directory are not supported. If multiple v5 policies reference the same file outside the `/policy` directory, the last applied policy will overwrite the file and will be used by all policies that reference it. Removing any policy that contains the file will cause the file to be deleted and may cause errors for the remaining policies that use it.
- If an extension contains or references objects and files that are also referenced by other objects outside of the extension, deleting the extension can cause errors that are difficult to troubleshoot.
- It is possible to delete an extension that is referenced by published APIs. This may cause those API requests to return errors.
- For an extension type of `user-defined-policy`, extra assembly functions may be present in the DataPower import `.zip` file, even if there is no `.yaml` file to indicate their presence. Using these functions without declaring them using a `.yaml` file is not supported. A policy `.yaml` file is required for each policy you want to advertise to API Manager.
- Extension type `user-defined-policy-yaml` may leak generated Gatewayscript `.js` files in temporary:/js/ or `.xslt` files when deleted.

- Extension types that use SOMA imports must use new objects with no overrides. If you override a singleton or shared object using an extension of this type, then the object will not be restored to its previous state when the extension is deleted, and those objects may be missing after the extension is deleted. You must then restore any needed objects.

- **Gateway extensions manifest**
  The gateway extensions manifest controls how gateway extensions are applied to the DataPower API Gateway.
- **v5 policy emulation limitations**
  This page details the limitations of v5 policy emulation for the current release.

Gateway extensions manifest

The gateway extensions manifest controls how gateway extensions are applied to the DataPower® API Gateway.

The gateway extensions manifest enables you to deploy multiple extension implementations in a single package. The manifest is a JSON file that lists the extensions `.zip` files to be applied and when they are to be deployed. You can also use the manifest to enable v5 policy emulation for the API Connect gateway service. V5 policy emulation enables you to deploy supported v5 built-in policies and custom policies, and manage them from API Manager without having to rewrite them for DataPower API Gateway. V5 policy emulation is supported only for policies that are applied to the Gateway service scope.

The `manifest.json` file must be at the root level of the extensions `.zip` file, along with the gateway extension files specified in the manifest. For example:

```
manifest.json
jws-sign-policy.zip
add_udp.zip
```

The `manifest.json` file contains the following sections.

`properties` section

The properties section determines whether to enable the v5 framework to deploy supported v5 built-in policies and custom policies and manage them from API Manager without having to rewrite them for DataPower API Gateway.

Table 1. `properties` section properties

Property	Description
`deploy-policy-emulator`	Determines whether to enable v5 framework emulation. Valid values are `true` and `false`. v5 framework emulation is enabled for the API Connect Gateway service if `deploy-policy-emulator` set to `true`, if a v5 policy is detected in the files list, or if a `deploy-policies` entry is specified.
`deploy-policies`	Contains a list of built-in policies to advertise for v5 emulation. The following policies are supported. • `activity-log_1.5.0` • `gatewayscript_1.0.0` • `if_1.5.0` • `invoke_1.5.0` • `proxy_1.5.0` • `redact_1.5.0` • `switch_1.5.0` • `validate-usernametoken_1.0.0` • `xslt_1.0.0`
`defer-all-override`	An override toggle to force the deployment property of all extension entries (including V5E) to be deferred. When enabled, the user must restart the `apic-gw-service` object to apply any extension events.

`files` section

The `files` section is an array of file entries that reference extension implementations. Each entry must reference a file that is in the `extension.zip` file (or the extension is rejected).

Table 2. `files` section properties

Property	Description
`filename`	The name of the file that contains the extension implementation. This property is required and cannot be changed while the extension is deployed as part of an update.

Property	Description
`type`	The type of extension implementation to be deployed. This property is required. The following types are supported. **`dp-import`** A DataPower import used to add additional configuration and/or services to a gateway. Like the extension type, but `dp-import` persists the configuration on physical or virtual DataPowers. Note: A `dp-import` file cannot be used to change the `apic-gw-service` object. If it is configured for this purpose, the extension is rejected. **`extension`** A legacy API Gateway extension that can be used to add additional configuration to a gateway. `Extension` should not be used to modify objects created by the API Connect gateway service, see gwd_extension for that use case. For more information, see Gateway extension guidelines - DataPower API Gateway. **`gwd_extension`** A JSON file that allows users to customize objects created by the API Connect gateway service. Only one `gwd_extension` is allowed in the manifest and its deployment is always deferred. For more information about the `gwd_extension`, see GWD extension type. **`policy-v5`** A complete, valid v5 policy implementation. For more information, see Authoring policies for the DataPower Gateway (v5 compatible). **`user-defined-policy`** A native API Gateway policy packaged in a DataPower import zip. Using a DataPower import to apply the policy implementations allows the configuration to persist through restarts and upgrades on virtual and physical forms. The import might contain one or more policy implementations. This policy type is always global scoped. For more information, see Defining, packaging, and publishing a global-scoped policy for the API Gateway. **`user-defined-policy-yaml`** A YAML file that includes an assembly. This policy type is always global scope. If the assembly references another user-defined policy, that policy must already be advertised to the API Manager before it can be deployed. For more information, see Working with global policies. **`v5_extension`** A migrated v5 extension that the migration utility generates. Only certain parts of v5 extensions are usable in API Gateway. For more information, see Migrating gateway extensions. `v5_extension` is reserved for v5 extensions that the migration utility creates. Do not manually create this type of extension. Note: V5E is intended to allow users to migrate from V5 without editing their APIs. V5E type policies cause known performances degradations in the API gateway. To improve performance of V5E policies, it is recommended to convert them to native policies.
`deploy`	When to deploy the extension entry. If you are deploying extensions that might cause a temporary API Connect gateway service outage, it is recommended to use deferred. **`immediate`** Deploy the extension entry immediately when the API Connect gateway service receives it. All gateway nodes apply the extension in parallel. Warning: `immediate` can cause an outage across the cluster, dependent on what the extension does. **`deferred`** A managed deployment strategy that applies the extension entry after the API Connect gateway service is restarted. This value is the default. Using deferred allows the user to do a controlled roll out of the extension across the cluster. When a DRR occurs this property is ignored, and all entries are deployed immediately.
`catalog`	Indicates that a v5 policy is to be used only by certain catalogs and should not be advertised as a gateway service-level user policy. This property is valid only if the file type is `policy-v5`. The `catalog` value must be in the format ***`orgname_catalogname`***.

Example `manifest.json`

The following example `manifest.json` file enables v5 policy emulation and specifies two built-in v5 policies to deploy. The files section lists two v5 policy implementation files and two extension files.

```
{
   "extension":{
      "properties":{
         "deploy-policy-emulator":true,
         "deploy-policies":[
            "invoke_1.5.0"
         ]
      },
      "files":[
         {
            "filename":"proxy_service.zip",
            "deploy":"immediate",
            "type":"dp-import"
         },
         {
            "filename":"add_udp.zip",
            "deploy":"deferred",
            "type":"extension"
         },
         {
            "filename":"override-parse-defaults.json",
            "type":"gwd_extension"
         },
         {
            "filename":"jws-sign-policy.zip",
            "deploy":"immediate",
            "type":"policy-v5"
         },
         {
            "filename":"jwe-decrypt-policy.zip",
            "deploy":"deferred",
            "type":"policy-v5",
```

```
            "catalog":"extension-org_ext-cat-2"
         },
         {
            "filename":"mypolicy.zip",
            "deploy":"immediate",
            "type":"user-defined-policy"
         },
         {
            "filename":"Invoke-Private-Backend-policy.yaml",
            "deploy":"immediate",
            "type":"user-defined-policy-yaml"
         }
      ]
   }
}
```

GWD extension type

The **gwd_extension** type is a JSON map that is used to apply user customizations to objects generated by the API Connect gateway service.

The **gwd_extension** map contains an array of user-defined CLI commands to be run on one or more objects. Only one **gwd_extension** is allowed in the manifest and its deployment is always deferred. Restarting the API Connect gateway service object is required to apply, update, or remove the overrides.

Key features of the **gwd_extension** type:

- The **gwd_extension** type can be used to define overrides at various scope-levels, which means a single array of override commands can be applied to one or more objects of a particular type.
- With some object mappings, you can define instances within a scope so that you can apply the custom CLI commands to all objects in that scope that match a particular name or criteria. For example, you can apply overrides to all catalog instances with the name **MyCatalog** regardless of which organization it belongs to.
- By default, the overrides are applied in order of scope granularity (global → organization → catalog → product → instance).
- Add **"@exclusive": true** to the object at any scope level to signal not to apply the previous scopes. Similarly, using an empty override array in a scope is a signal to not apply all previous scope overrides for that mapping, see tls-client for an example.

The supported scopes are:

- **_global**: Applies customizations to all objects of a particular type. For example, add one or more custom assembly-rate-limits to all **api-collections**.
  - **_instance**: Applies customizations to all objects that match a name or criteria. For example, add one or more custom **assembly-rate-limits** to all **api-collections** with the name **my-restricted-catalog** regardless which organization it belongs to.
- **_organization**: Applies customizations to all objects used in an organization. For example, apply a customization to all TLS client profiles that are used in the organization called **provorg**.
  - **_instance**: Applies customizations to all objects that match a name or criteria. For example, applying a customization to all TLS client profiles with a particular name and version regardless of which catalog is using it.
- **_catalog**: Applies customizations to all objects used in a catalog. For example, apply a customization to all TLS client profiles used in the catalog **myCat** of organization **myOrg**.
  - **_instance**: Applies customizations to all objects that match a name or criteria. For example, applying a customization to all plans in a catalog with a particular name regardless of which product it comes from.
- **_product**: Applies customizations to all objects of a type used in a product. For example, apply a customization to all plans in **product my-prod:1.0.0** published in catalog **myCat** of organization **myOrg**. The product scope is only used by the api-plan **gwd_extension** extension type.
- **_instance**: Applies customizations to a single instance. For example, apply a customization to TLS client profile **my-profile:1.0.0** in catalog **myCat** of organization **myOrg**.

Example **gwd_extension** that shows multiple types of overrides:

```
{
  "parse-settings":{
     "_global":{
        "override":[
           "document-size 4194304"
        ]
     }
  },
  "apigw":{
     "_global":{
        "override":[
           "assembly-rate-limit ratelimit-reject-ALL 10000 5 second on on on off off off \"\" \"1\"",
           "assembly-rate-limit ratelimit-alert-ALL 7500 5 second off on on off off off \"\" \"1\""
        ]
     }
  }
}
```

The **gwd_extension** type supports the following objects:

- **config-sequence**
  The config-sequence object created by **apic-gw-service**, which is used to create all API gateway objects.
- **parse-settings**
  Parse settings used by the API Gateway.
- **analytics-endpoint**
- **front-side-handler**
  FSH and associated SNI Server profiles and objects that are used by the API Gateway.
- **apigw**
  API Gateway object.
- **tls-client**

- **api-collection**
  Catalog objects.
- **api-plan**
  API Plan objects that are created from published products.
- **registry**
  Registry objects used in a catalog.
- **oauth-provider**
  Oauth objects used in a catalog.
- **api-definition**
  The API objects that are published to one or more catalogs.
- **assembly**
  The assembly objects and their associated actions that are used by APIs and global policies.

config-sequence

The config-sequence object created by **apic-gw-service**, which is used to create all API gateway objects.

- Override Key: override
- Scopes:
  - _global
- Example:

```
{
   "config-sequence":{
      "_global":{
         "override":[
            "summary my-override"
         ]
      }
   }
}
```

parse-settings

Parse settings used by the API Gateway.

- Override Key: override
- Scopes:
  - _global
- Example:

```
{
   "parse-settings":{
      "_global":{
         "override":[
            "summary my-override"
         ]
      }
   }
}
```

analytics-endpoint

- Override Key: override
- Scopes:
  - _global
- Example:

```
{
   "analytics-endpoint":{
      "_global":{
         "override":[
            "summary my-override"
         ]
      }
   }
}
```

front-side-handler

FSH and associated SNI Server profiles and objects that are used by the API Gateway.

- Override Key: `source-https`
- Child Objects:
  - `ssl-sni-server`

    - `ssl-sni-map`
    - `ssl-server`
        - Override Key: profile
        - _instance: SNI hostname key
        - Child Objects:
            - `valcred`
            - `idcred`
- Scopes:
    - _global
- Example:

```
{
   "front-side-handler":{
      "_global":{
         "source-https":[
            "summary global my-fsh-override"
         ],
         "ssl-sni-server":[
            "summary global my-sni-server-override"
         ],
         "ssl-sni-map":[
            "summary global my-sni-map-override"
         ],
         "ssl-server":{
            "_instance":{
               "*.myhost.com":{
                  "profile":[
                     "summary sni-host-instance-my-profile-override"
                  ],
                  "valcred":[
                     "summary sni-host-instance-my-valcred-override"
                  ],
                  "idcred":[
                     "summary sni-host-instance-my-idcread-override"
                  ]
               }
            },
            "profile":[
               "summary my-profile-override"
            ],
            "valcred":[
               "summary my-valcred-override"
            ],
            "idcred":[
               "summary my-idcread-override"
            ]
         }
      }
   }
}
```

apigw

API Gateway object.

- Override Key: override
- Scopes:
    - _global
- Example:

```
{
   "apigw":{
      "_global":{
         "override":[
            "summary my-override"
         ]
      }
   }
}
```

tls-client

Profiles that are attached to catalogs.

- Override Key: profile
- Child Objects:
    - Valcred
    - Idcred
- Scopes:
    - _global
        - _instance: *<profileName>*V*<profileVersion>*
    - _organization
        - _instance: *<profileName>*V*<profileVersion>*

    - _catalog
    - _instance: *<profileName>*V*<profileVersion>*
- Example: In this example, an empty array for **valcred** in catalog scope is used. The use of any empty array means the **valcreds** for TLS profiles in that catalog do not get the global or organizational overrides.

```
{
   "tls-client":{
      "_global":{
         "_instance":{
            "my-profileV1.0.0":{
               "profile":[
                  "summary global-instance-profile-override"
               ],
               "valcred":[
                  "summary global-instance-valcred-override"
               ],
               "idcred":[
                  "summary global-instance-idcred-override"
               ]
            }
         },
         "profile":[
            "summary global-profile-override"
         ],
         "valcred":[
            "summary global-valcred-override"
         ],
         "idcred":[
            "summary global-valcred-override"
         ]
      },
      "_organization":{
         "my-org":{
            "profile":[
               "summary org-profile-override"
            ]
         }
      },
      "_catalog":{
         "my-org":{
            "my-catalog":{
               "profile":[
                  "summary cat-profile-override"
               ],
               "valcred":[

               ]
            }
         }
      },
      "_instance":{
         "my-org":{
            "my-catalog":{
               "my-tlsV1.0.0":{
                  "profile":[
                     "summary instance-profile-override"
                  ],
                  "valcred":[
                     "summary instance-valcred-override"
                  ]
               }
            }
         }
      }
   }
}
```

api-collection

Catalog objects.

- Override Key: collection
- Child Objects:
    - **preflow-assembly-function**
    - **preflow-assembly-function-call**
    - **postflow-assembly-function**
    - **postflow-assembly-function-call**
    - **global-assembly-rule**
    - **global-assembly**
    - **global-assembly-function**
    - **global-assembly-function-call**
    - **assembly-function-rule**
- Scopes:
    - _global
        - _instance: *<catalogName>*
    - _organization
    - _catalog

- Example: In the example, global instance is used to apply override to all `api-collections` that have the name `my-catalog`, regardless of what organization they belong to.

```
{
   "api-collection":{
      "_global":{
         "_instance":{
            "my-catalog":{
               "collection":[
                  "summary hello my-catalog"
               ]
            }
         },
         "preflow-assembly-function":[
            "summary hello preflow-assembly-function global scope"
         ],
         "preflow-assembly-function-call":[
            "summary hello preflow-assembly-function-call global scope"
         ],
         "postflow-assembly-function":[
            "summary hello postflow-assembly-function global scope"
         ],
         "postflow-assembly-function-call":[
            "summary hello postflow-assembly-function-call global scope"
         ],
         "global-assembly-rule":[
            "summary hello global-assembly-rule global scope"
         ],
         "global-assembly":[
            "summary hello global-assembly global scope"
         ],
         "global-assembly-function":[
            "summary hello global-assembly-function global scope"
         ],
         "global-assembly-function-call":[
            "summary hello global-assembly-function-call global scope"
         ],
         "assembly-function-rule":[
            "summary hello assembly-function-rule global scope"
         ],
         "collection":[
            "summary hello collection global scope"
         ]
      },
      "_organization":{
         "my-org":{
            "collection":[
               "summary hello collection org scope"
            ]
         }
      },
      "_catalog":{
         "my-org":{
            "my-catalog":{
               "collection":[
                  "summary hello collection cat scope"
               ]
            }
         }
      }
   }
}
```

api-plan

API Plan objects that are created from published products.

- Override Key: override
- Child Objects:
  - `graphql-schema-options`
    - Override Key: override
    - Child Objects:
      - `visibility-list`
  - `rate-limit`
    - Child Object:
      - `rate-limit-definition`
        - Override Key: override
  - `burst-limit`
    - Child Object:
      - `rate-limit-definition`
        - Override Key: override
  - `rate-limit-group`
    - Override Key: override
  - `api-operation-rate-limit`
    - Override Key: override
    - Child Objects:
      - `rate-limit-group`

        - `rate-limit-definition`
    - `assembly-limit`
        - Child Objects:
            - `rate-limit`
            - `burst-limit`
            - `count-limit`
- Scopes:
    - global
        - _instance: *<planName>*.
    - organization
        - _instance: *<planName>*.
    - catalog
        - _instance: *<planName>*.
    - product: *<name>*:*<version>*
    - _instance: *<planName>*.
- Example:

```
{
   "api-plan":{
      "_global":{
         "graphql-schema-options":{
            "visibility-list":[
               "echo hello graphql-schema-options visibility _global"
            ],
            "override":[
               "echo hello graphql-schema-options _global"
            ]
         },
         "rate-limit":{
            "rate-limit-definition":{
               "override":[
                  "echo hello rate-limit _global"
               ]
            }
         },
         "burst-limit":{
            "rate-limit-definition":{
               "override":[
                  "echo hello burst-limit _global"
               ]
            }
         },
         "rate-limit-group":{
            "override":[
               "echo hello rate-limit-group _global"
            ]
         },
         "api-operation-rate-limit":{
            "rate-limit-definition":{
               "override":[
                  "echo hello api-operation-rate-limit _global"
               ]
            },
            "rate-limit-group":{
               "override":[
                  "echo hello api-operation-rate-limit-group _global"
               ]
            },
            "override":[
               "echo hello api-operation-rate-limit _global"
            ]
         },
         "assembly-limit":{
            "rate-limit":{
               "rate-limit-definition":{
                  "override":[
                     "echo hello assemby rate-limit _global"
                  ]
               }
            },
            "burst-limit":{
               "rate-limit-definition":{
                  "override":[
                     "echo hello assebmly burst-limit _global"
                  ]
               }
            },
            "count-limit":{
               "rate-limit-definition":{
                  "override":[
                     "echo hello assebmly count-limit _global"
                  ]
               }
            }
         },
         "override":[
            "echo hello api-plan _global"
         ]
      },
      "_organization":{
         "my-org":{
            "override":[
```

```
            "summary hello api-plan _organization"
         ]
      }
   },
   "_catalog":{
      "my-org":{
         "my-catalog":{
            "override":[
               "summary hello api-plan _catalog"
            ]
         }
      }
   },
   "_product":{
      "my-org-2":{
         "my-cat-2":{
            "my-product:1.0.0":{
               "override":[
                  "summary hello api-plan _product"
               ]
            }
         }
      }
   },
   "_instance":{
      "my-org3":{
         "my-catalog-3":{
            "my-product:1.0.0":{
               "my-plan-1":{
                  "override":[
                     "summary hello api-plan _instance"
                  ]
               }
            }
         }
      }
   }
}
```

registry

Registry objects used in a catalog.

- Registry types:
  - LDAP
    - Override Key: override
    - Child object: `ldap-search-parameters`
  - AuthUrl
    - Override Key: override
- Scopes:
  - _global
  - _organization
  - _catalog
  - _instance: APIM name
- Example:

```
{
   "registry":{
      "ldap":{
         "_global":{
            "override":[
               "ldap-readtimeout 120"
            ],
            "ldap-search-parameters":[
               "summary override parameters global"
            ],
            "_instance":{
               "my-ldap-registry":{
                  "ldap-search-parameters":[
                     "summary override parameters global instance"
                  ],
                  "override":[
                     "ldap-port 636",
                     "ssl-client ldap-ssl-client"
                  ]
               }
            }
         },
         "_organization":{
            "my-org":{
               "override":[

               ],
               "_instance":{
                  "my-ldap-registry":{
                     "override":[
                        "ldap-version v3",
                        "ldap-bind-password mypw",
```

```
                        "ldap-search-param ldap-param",
                        "summary override org instance"
                     ]
                  }
               }
            }
         },
         "_catalog":{
            "my-org":{
               "my-catalog":{
                  "override":[
                     "ldap-host 1.2.3.4",
                     "ldap-bind-dn cn=root,dc=datapower,dc=com",
                     "summary override catalog"
                  ]
               }
            }
         },
         "_instance":{
            "my-org-3":{
               "my-catalog-3":{
                  "my-ldap-reg":{
                     "override":[

                     ],
                     "ldap-search-parameters":[

                     ]
                  }
               }
            }
         }
      },
      "authUrl":{
         "_global":{
            "override":[
               "summary override global"
            ],
            "_instance":{
               "httpbin":{
                  "override":[
                     "summary override global instance"
                  ]
               }
            }
         },
         "_organization":{
            "my-org":{
               "override":[

               ],
               "_instance":{
                  "httpbin":{
                     "override":[
                        "summary override org instance"
                     ]
                  }
               }
            }
         },
         "_catalog":{
            "my-org":{
               "my-catalog":{
                  "_instance":{
                     "httpbin":{
                        "override":[
                           "summary override cat instance",
                           "echo some other override"
                        ]
                     }
                  },
                  "override":[
                     "summary override cat"
                  ]
               }
            }
         },
         "_instance":{
            "my-org-3":{
               "my-catalog-3":{
                  "my-authurl":{
                     "override":[
                        summary override instance
                     ]
                  }
               }
            }
         }
      }
   }
}
```

oauth-provider

Oauth objects used in a catalog.

- Oauth Types
  - Native
    - Override Key: override
  - Third-party
    - Override Key: override
- Scopes:
  - _global
    - _instance: APIM name
  - _organization
    - _instance: APIM name
  - _catalog
  - _instance: APIM name
- Example

```
{
   "oauth-provider":{
      "native":{
         "_global":{
            "override":[
               "apic-access-token-ttl 86400"
            ],
            "_instance":{
               "my-native-oauth":{
                  "override":[
                     "scopes-allowed cmcscope1",
                     "enable-debug-mode on"
                  ]
               }
            }
         },
         "_organization":{
            "my-org-2":{
               "override":[

               ]
            }
         },
         "_instance":{
            "my-org":{
               "my-catalog":{
                  "my-native-oauth-2":{
                     "override":[
                        "apic-enable-introspection on",
                        "apic-auth-code-ttl 600",
                        "apic-enable-refresh-token on "
                     ]
                  }
               }
            }
         }
      },
      "third-party":{
         "_organization":{
            "my-org":{
               "_instance":{
                  "my-third-party-oauth":{
                     "override":[
                        "third-party-introspect-url https://someaddress.com"
                     ]
                  }
               }
            }
         }
      }
   }
}
```

api-definition

The API objects that are published to one or more catalogs.

- Override Key: api-definition
- Child Objects
  - api-schema
  - Security-definition
    - oauth2
      - api-sec-oauth
      - _instance: swagger name
    - apiKey
      - api-sec-apikey
      - _instance: swagger name
    - http
      - api-sec-http-scheme
      - _instance: swagger name

        - basic
            - x-ibm-authentication-url
            - api-sec-basic-auth
            - _instance: swagger name
    - security-requirements
        - api-sec-oauth-req
            - Override Key: override
            - _instance: Datapower object name
        - api-sec-req
    - api-path
        - Override Key: path
        - _instance: *<Path URI>*
        - Child Objects
            - api-operation
                - Override Key: operation
                - _instance: *<Operation Method>*
                - Child Objects
                    - security-requirement
                        - api-sec-oauth-req
                            - Override Key: override
                            - _instance: Datapower object name
                        - api-sec-req
                    - api-schema-request
                    - api-schema-response
                        - Override Key: override
                        - _instance: Datapower object name
                - api-schema-request
    - wsdl-validation-schema-location
    - cors-policy
    - cors-rule
        - Override Key: override
        - _instance: Datapower object name
- Scopes:
    - _global
        - _instance: name:version
    - _organization
        - _instance: name:version
    - _catalog
    - _instance: name:version
- Example: In this example, the **`@exclusive`** flag is used on the instance nick-limits:1.0.0 in my-catalog of my-organization. The **`@exclusive`** flag means that only the CLI commands at the instance level are applied for that API while all other APIs in that catalog still have the global, organization, and catalog CLI commands applied to them.

```
{
   "api-definition":{
      "_global":{
         "api-schema":[
            "echo hello schema global scope"
         ],
         "security-definition":{
            "oauth2":{
               "api-sec-oauth":[
                  "echo hello oauth def global scope"
               ]
            },
            "apiKey":{
               "api-sec-apikey":[
                  "echo hello apikey global scope"
               ]
            },
            "http":{
               "api-sec-http-scheme":[
                  "echo hello http scheme global scope"
               ]
            },
            "basic":{
               "x-ibm-authentication-url":[
                  "echo hello ibm auth global scope"
               ],
               "api-sec-basic-auth":[
                  "echo hello basic auth global scope"
               ]
            }
         },
         "security-requirements":{
            "api-sec-oauth-req":{
               "override":[
                  "echo hello oauth req global scope"
               ]
            },
            "api-sec-req":[
               "echo hello seq req global scope"
            ]
         },
         "api-path":{
            "_instance":{
               "/new-path-1":{
```

```
                "api-schema-request":[
                   "echo hello path instance /new-path-1 request schema global scope"
                ],
                "path":[
                   "echo hello path instance /new-path-1 path global scope"
                ]
             }
          },
          "api-operation":{
             "_method":{
                "GET":{
                   "security-requirements":{
                      "api-sec-oauth-req":{
                         "_instance":{

                         },
                         "override":[
                            "echo hello GET api-sec-oauth-req global scope"
                         ]
                      },
                      "api-sec-req":[
                         "echo hello GET api-sec-req global scope"
                      ]
                   },
                   "api-schema-request":[
                      "echo hello GET api-schema-request global scope"
                   ],
                   "api-schema-response":{
                      "_instance":{

                      },
                      "override":[
                         "echo hello GET api-schema-response global scope"
                      ]
                   },
                   "operation":[
                      "echo hello GET operation global scope"
                   ]
                }
             },
             "security-requirements":{
                "api-sec-oauth-req":{
                   "override":[
                      "echo hello oauth req global scope"
                   ]
                },
                "api-sec-req":[
                   "echo hello seq req global scope"
                ]
             },
             "api-schema-request":[
                "echo hello operation request schema"
             ],
             "api-schema-response":{
                "override":[
                   "echo hello operation response schema global scope"
                ]
             },
             "operation":[
                "echo hello operation global scope"
             ]
          },
          "api-schema-request":[
             "echo hello request schema global scope"
          ],
          "path":[
             "echo hello path global scope"
          ]
       },
       "wsdl-validation-schema-location":[
          "echo hello wsdl-val global scope"
       ],
       "cors-policy":[
          "echo hello cors-policy global scope"
       ],
       "cors-rule":{
          "override":[
             "echo hello cors-rule global scope"
          ]
       },
       "api-definition":[
          "echo hello api-def global scope"
       ]
    },
    "_organization":{
       "my-organization":{
          "api-definition":[
             "echo hello api-def _organization scope"
          ]
       }
    },
    "_catalog":{
       "my-organization":{
          "my-catalog":{
             "api-definition":[
                "echo hello api-def _catalog scope"
```

```
            ]
         }
      }
   },
   "_instance":{
      "my-organization":{
         "my-catalog":{
            "nick-limits:1.0.0":{
               "@exclusive":true,
               "api-schema":[
                  "echo hello schema _instance scope"
               ],
               "security-definition":{
                  "oauth2":{
                     "api-sec-oauth":[
                        "echo hello oauth def _instance scope"
                     ]
                  },
                  "apiKey":{
                     "api-sec-apikey":[
                        "echo hello apikey _instance scope"
                     ]
                  },
                  "http":{
                     "api-sec-http-scheme":[
                        "echo hello http scheme _instance scope"
                     ]
                  },
                  "basic":{
                     "x-ibm-authentication-url":[
                        "echo hello ibm auth _instance scope"
                     ],
                     "api-sec-basic-auth":[
                        "echo hello basic auth _instance scope"
                     ]
                  }
               },
               "security-requirements":{
                  "api-sec-oauth-req":{
                     "override":[
                        "echo hello oauth req _instance scope"
                     ]
                  },
                  "api-sec-req":[
                     "echo hello seq req _instance scope"
                  ]
               },
               "api-path":{
                  "_instance":{
                     "/new-path-2":{
                        "api-schema-request":[
                           "echo hello path instance request schema _instance scope"
                        ],
                        "path":[
                           "echo hello path instance / path _instance scope"
                        ]
                     }
                  },
                  "api-operation":{
                     "_method":{
                        "GET":{
                           "security-requirements":{
                              "api-sec-oauth-req":{
                                 "override":[
                                    "echo hello GET api-sec-oauth-req _instance scope"
                                 ]
                              },
                              "api-sec-req":[
                                 "echo hello GET api-sec-req _instance scope"
                              ]
                           },
                           "api-schema-request":[
                              "echo hello GET api-schema-request _instance scope"
                           ],
                           "api-schema-response":{
                              "override":[
                                 "echo hello GET api-schema-response _instance scope"
                              ]
                           },
                           "operation":[
                              "echo hello GET operation _instance scope"
                           ]
                        },
                        "security-requirements":{
                           "api-sec-oauth-req":{
                              "override":[
                                 "echo hello oauth req _instance scope"
                              ]
                           },
                           "api-sec-req":[
                              "echo hello seq req _instance scope"
                           ]
                        },
                        "api-schema-request":[
                           "echo hello operation request schema"
                        ],
```

```
                        "api-schema-response":{
                           "override":[
                              "echo hello operation response schema _instance scope"
                           ]
                        },
                        "operation":[
                           "echo hello operation _instance scope"
                        ]
                     }
                  },
                  "api-schema-request":[
                     "echo hello request schema _instance scope"
                  ],
                  "path":[
                     "echo hello path _instance scope"
                  ]
               }
            },
            "wsdl-validation-schema-location":[
               "echo hello wsdl-val _instance scope"
            ],
            "cors-policy":[
               "echo hello cors-policy _instance scope"
            ],
            "cors-rule":{
               "override":[
                  "echo hello cors-rule _instance scope"
               ]
            },
            "api-definition":[
               "echo hello api-def _instance scope"
            ]
         }
      }
   }
}
```

assembly

The assembly objects and their associated actions that are used by APIs and global policies.

- Override Key: assembly
- Child Objects
  - api-rule
  - assembly-action
    - activity-log_1.5.0
      - Override Key: override
      - _instance: Title or Datapower object name
    - client-identification
      - Override Key: override
      - _instance: Title or Datapower object name
    - client-security
      - Override Key: override
      - _instance: Title or Datapower object name
    - cors
      - Override Key: override
      - _instance: Title or Datapower object name
    - extract
      - Override Key: override
      - _instance: Title or Datapower object name
    - function-call
      - Override Key: override
      - _instance: Title or Datapower object name
    - gatewayscript
      - Override Key: override
      - _instance: Title or Datapower object name
    - graphql-execute
      - Override Key: override
      - _instance: Title or Datapower object name
      - Child Objects
        - assembly
          - Override Key: assembly
          - Child Objects
            - api-rule
            - error-rule
              - Override Key: api-rule
              - _instance: Error Name or Datapower object name
            - default
            - finally
        - target
          - _instance: Title or Datapower object name
          - Child Objects

        - assembly-function
        - assembly-function-call
- graphql-introspect
    - Override Key: override
    - _instance: Title or Datapower object name
- "html-page
    - Override Key: override
    - _instance: Title or Datapower object name
- if
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - api-rule
        - assembly-switch
- if_1.5.0
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - assembly-gatewayscript
        - switch-api-rule
        - assembly-switch
        - api-rule
        - assembly-function-call
- invoke
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - header-control
        - parameter-control
- json-to-xml
    - Override Key: override
    - _instance: Title or Datapower object name
- jwt-generate
    - Override Key: override
    - _instance: Title or Datapower object name
- jwt-validate
    - Override Key: override
    - _instance: Title or Datapower object name
- log
    - Override Key: override
    - _instance: Title or Datapower object name
- map
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - parse-override
- oauth
    - Override Key: override
    - _instance: Title or Datapower object name
- operation-switch
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - otherwise-api-rule
        - api-rule
        - condition-api-rules
            - Override Key: api-rule
            - _instance: *<Method>*:*<Path>* or Datapower object name
- parse
    - Override Key: override
    - _instance: Title or Datapower object name
    - Child Objects
        - parse-settings
- proxy
    - Override Key: override
    - _instance: Title or Datapower object name
- ratelimit
    - Override Key: override
    - _instance: Title or Datapower object name
- redact
    - Override Key: override
    - _instance: Title or Datapower object name
- redact_1.5.0
    - Override Key: override
    - _instance: Title or Datapower object name
- result
    - Override Key: override
    - _instance: Title or Datapower object name
- validate
    - Override Key: override

        - _instance: Title or Datapower object name
        - Child Objects
            - api-schema
    - security
        - Override Key: override
        - _instance: Title or Datapower object name
    - switch
        - Override Key: override
        - _instance: Title or Datapower object name
        - Child Objects
            - otherwise-api-rule
            - condition-api-rules
                - Override Key: api-rule
                - _instance: Condition Statement or Datapower object name
    - switch_1.5.0
        - Override Key: override
        - _instance: Title or Datapower object name
        - Child Objects
            - assembly-gatewayscript
            - otherwise-api-rule
            - condition-api-rules
                - Override Key: api-rule
                - _instance: Condition Statement or Datapower object name
    - throw
        - Override Key: override
        - _instance: Title or Datapower object name
    - user-security
        - Override Key: override
        - _instance: Title or Datapower object name
        - Child Objects
            - auth-url
    - websocket-upgrade
        - Override Key: override
        - _instance: Title or Datapower object name
        - Child Objects
            - header-control
            - parameter-control
            - websocket-request
                - Override Key: assembly
                - Child Objects
                    - api-rule
                    - error-rule
                        - Override Key: api-rule
                        - _instance: Error Name or Datapower object name
                    - default
                    - finally
            - websocket-response
                - Override Key: assembly
                - Child Objects
                    - api-rule
                    - error-rule
                        - Override Key: api-rule
                        - _instance: Error Name or Datapower object name
                    - default
                    - finally
    - wsdl
        - Override Key: override
        - _instance: Title or Datapower object name
    - xml-to-json
        - Override Key: override
        - _instance: Title or Datapower object name
- error-rule
    - Override Key: api-rule
    - _instance: Error Name
- default
- finally

- Scopes:
    - _global
        - _instance: swagger name:version
    - _organization
        - _instance: swagger name:version
    - _catalog
    - _instance: swagger name:version
- Example: In this example, 2 instances are included within the global scope mapping. First is, **`my-global-policy:1.0.0`**, which is a global-policy that is used in some catalogs. Since it is defined at the global scope, the overrides apply anytime an assembly is built from a swagger with the name and version of **`my-global-policy:1.0.0`**. Second is, **`complex-switch:1.0.0`**, which is an API that is published to various catalogs. Similarly, since it is defined as an instance with in the global scope the overrides apply anytime that the assembly is built from a swagger with the name and version of **`complex-switch:1.0.0`**. Also see the **`set-var`** example within global scope for an example of overriding assembly action instances based on the title that is given in APIM or the DataPower object name.

```
{
   "assembly":{
      "_global":{
         "_instance":{
            "my-global-policy:1.0.0":{
               "api-rule":[
                  "echo hello global-policy api rule at the global instance scope"
               ],
               "assembly":[
                  "echo hello global-policy assembly at the global instance scope"
               ],
               "error-rule":{
                  "_instance":{
                     "ConnectionError":{
                        "api-rule":[
                           "echo hello global-policy  assembly error rule override global instance with rule instance
scope"
                        ]
                     }
                  },
                  "api-rule":[
                     "echo hello global-policy assembly error rule override global instance scope"
                  ]
               },
               "default":[
                  "echo hello global-policy assembly default error rule override global instance scope"
               ],
               "finally":[
                  "echo hello global-policy assembly finally rule override global instance scope"
               ],
               "assembly-actions":{
                  "cors":{
                     "override":[
                        "echo hello global-policy cors global instance scope"
                     ]
                  },
                  "wsdl":{
                     "override":[
                        "echo hello global-policy wsdl global instance scope"
                     ]
                  },
                  "client-identification":{
                     "override":[
                        "echo hello global-policy client-identification global instance scope"
                     ]
                  },
                  "invoke":{
                     "_instance":{
                        "my-custom-invoke":{
                           "override":[
                              "echo hello global-policy invoke global instance with title instance scope"
                           ]
                        }
                     },
                     "override":[
                        "echo hello invoke global custom policy instance scope"
                     ]
                  }
               }
            },
            "complex-switch:1.0.0":{
               "api-rule":[
                  "echo hello api rule global api instance scope"
               ],
               "assembly":[
                  "echo hello api assembly global api instance scope"
               ],
               "error-rule":{
                  "api-rule":[
                     "echo hello api assembly error rule override global api instance  scope"
                  ]
               },
               "default":[
                  "echo hello api assembly default error rule override global api instance scope"
               ],
               "finally":[
                  "echo hello api assembly finally rule override global api instance scope"
               ],
               "assembly-actions":{
                  "switch":{
                     "otherwise-api-rule":[
                        "echo hello switch otherwise api-rule global scope api instance instance"
                     ],
                     "condition-api-rules":{
                        "_instance":{
                           "($statusCode() = 200)":{
                              "api-rule":[
                                 "echo hello switch api-rule global scope api instance instance"
                              ]
                           }
                        },
                        "api-rule":[
                           "echo hello switch api-rule global api instance scope"
                        ]
                     },
                     "override":[
```

```
                    "echo hello switch global api instance scope"
                 ]
              }
           }
        }
     },
     "api-rule":[
        "echo hello assembly api-rule global scope"
     ],
     "assembly":[
        "echo hello assembly global scope"
     ],
     "error-rule":{
        "api-rule":[
           "echo hello assembly error rule override global scope"
        ]
     },
     "default":[
        "echo hello assembly default error rule override global scope"
     ],
     "finally":[
        "echo hello assembly finally rule override global scope"
     ],
     "assembly-actions":{
        "set-variable":{
           "_instance":{
              "set-variable-delete":{
                 "override":[
                    "echo hello set var global instance scope using title"
                 ]
              },
              "addGatewayExtension-v6-org_addGatewayExtension-v6-catalog-140_complex-switch_1.0.0_set-variable_2":{
                 "override":[
                    "echo hello set var global instance scope using dp name"
                 ]
              }
           },
           "override":[
              "echo hello set var global scope"
           ]
        }
     }
  },
  "_organization":{
     "my-organization":{
        "api-rule":[
           "echo hello rule _organization scope"
        ],
        "assembly":[
           "echo hello assembly _organization scope"
        ],
        "assembly-actions":{
           "graphql-execute":{
              "targets":{
                 "assembly-function":[
                    "echo hello graphql-execute target af _organization scope"
                 ],
                 "assembly-function-call":[
                    "echo hello graphql-execute target afc _organization scope"
                 ]
              },
              "override":[
                 "echo hello graphql-execute _organization scope"
              ]
           }
        }
     }
  },
  "_catalog":{
     "my-organization":{
        "my-catalog":{
           "api-rule":[
              "echo hello rule _catalog scope"
           ],
           "assembly":[
              "echo hello assembly _catalog scope"
           ],
           "assembly-actions":{
              "if":{
                 "api-rule":[
                    "echo hello if api-rule _catalog scope"
                 ],
                 "assembly-switch":[
                    "echo hello if assembly-switch _catalog scope"
                 ]
              }
           }
        }
     }
  },
  "_instance":{
     "my-organization":{
        "my-catalog":{
           "complex-switch:1.0.0":{
              "api-rule":[
                 "echo hello rule _instance scope"
```

```
                ],
                "assembly":[
                   "echo hello assembly _instance scope"
                ],
                "assembly-actions":{
                   "invoke":{
                      "override":[
                         "echo hello invoke _instance scope"
                      ]
                   }
                }
             }
          }
       }
    }
}
```

API Gateway

v5 policy emulation limitations

This page details the limitations of v5 policy emulation for the current release.

- The reverse solidus character (\) might be removed from policy property values.
- The source parameter for XSLT and GatewayScript policies is base-64 encoded to preserve escaping. It cannot be easily viewed from the DataPower command line or the DataPower WebGUI.
- Some DataPower service variables (for example, **`var://service/mpgw/skip-backside`**) might not be applicable in API Gateway, depending on how the variables are used.
- User-defined policies that make any alterations to a DataPower context are not supported.
- Execution of a 1.0.0 policy may fail if a previous 2.0.0 policy updates the **`message.body`** content type (for example, from XML to JSON, or JSON to XML).
- A proxy 1.5.0 policy can return an incorrect response if it is the last action within a switch **`case`** or **`otherwise`** property. The response is parsed. The correct behavior is to not parse the response. To work around this issue, set the **`final-action:true`** parameter on the proxy.
- Proxy 1.5.0 and invoke 1.5.0 policies do not preserve the **`gtid`** header.
- The **`cache-putpost-response`** property is not included in the schema for invoke 1.5.0 and proxy 1.5.0 policies.
- Setting HTTP response headers in a v5 user-defined policy, GatewayScript policy, or XSLT policy is not supported.
- Non-UTF-8 characters are not supported.
- JSON data returned in the response of an API running in v10 with v5 emulation is formatted without spaces.
- The v5 user-defined policy emulation feature creates v5 compatible context. After running the user-defined policy, the emulation feature analyzes the context and converts it back to DataPower® API Gateway compatible context. The computations involved in this process might negatively impact the DataPower API Gateway performance. For better performance, use a non-emulated user-defined policy. For more information, see Authoring Policies for DataPower API Gateway.

Configuring your Gateway server extensions

You can add extra IBM® DataPower® enforcement capabilities to a Gateway service by uploading a .zip file that defines the required extension behavior, and then enabling the extension in IBM DataPower.

About this task

You can upload a Gateway server extension to a Gateway service in either of the following ways:

- Use the Cloud Manager user interface
- Use the developer toolkit CLI

Procedure

- To upload a Gateway server extension to a Gateway service by using the Cloud Manager user interface, complete the following steps:
  1. In the Cloud Manager user interface, click Topology.
  2. Alongside the required Gateway service, click the options icon ⋮, then click Configure gateway extension.
  3. Click Add.
     Note: You cannot upload more than one Gateway extension .zip file to the same Gateway service. If a Gateway server extension has previously been uploaded, the Add button is not available; you can replace the previous extension by clicking the options icon ⋮ alongside gateway-extension, then clicking Edit.
  4. Upload the extension as indicated, either by dragging and dropping the .zip file, or by browsing and selecting it.
  5. Click Save when done.
- To upload a Gateway server extension by using the developer toolkit CLI, complete the following steps:
  1. Log in to your API Connect Cloud Manager server as an administrator, using the following command:

     `apic login --server <cloud-manager_endpoint_url> --username <admin_user_ID> --password <admin_password> --realm admin/<identity_provider>`

     where:
     - *<cloud-manager_endpoint_url>* is the Cloud Manager endpoint URL; for example: **`mgmt-admin.<myserver>.com`**.
     - *<admin_user_ID>* is the user ID of your administrator account, and is the same as the user ID that you use to log in to the Cloud Manager user interface.
     - *<admin_password>* is the password for your administrator account.

- *<identity_provider>* is the identity provider that is used to authenticate administrative users.

For example:

```
apic login --server mgmt-admin.myserver.com --username admin --password my-admin-password --realm admin/myldap
```

2. Upload the Gateway extension .zip file by using the following command:

```
apic gateway-extensions:create <extension_zip_file> --scope org --org admin --gateway-service <gateway_service> --
availability-zone <availability-zone> --server <cloud-manager_endpoint_url>
```

where:

- *<extension_zip_file>* is the Gateway extension .zip file that you want to upload.
- *<gateway_service>* is the name of the Gateway service that you want add the extension to.
- *<availability-zone>* is the name of the availability zone that contains the Gateway service.
- *<cloud-manager_endpoint_url>* is the Cloud Manager endpoint URL.

For example:

```
apic gateway-extensions:create myextension.zip --scope org --org admin --gateway-service mygatewayservice --
availability-zone availability-zone-default --server mgt-admin.myserver.com
```

This example uses the default supplied availability zone name of `availability-zone-default`, which will be the required value if you have not configured your own availability zones. To check the names of the currently configured availability zones, use the following command:

```
apic availability-zones:list --org admin --server <cloud-manager_endpoint_url>
```

For details on configuring availability zones, see Creating an availability zone.
To check the names of the currently configured gateway services, use the following command:

```
apic gateway-services:list --org admin --availability-zone <availability-zone> --server <cloud-manager_endpoint_url>
```

You can confirm that the extension has been added to the Gateway service by using the `gateway-extensions:get` command; for example:

```
apic gateway-extensions:get --scope org --org admin --gateway-service mygatewayservice --availability-zone
availability-zone-default --server mgt-admin.myserver.com --output -
```

(the parameter setting `--output -` writes the details of the Gateway extension object to the command window. You can specify the name of an existing folder to have the details written to a `.yaml` file in that folder.)
For reference details of the `apic gateway-extensions` commands, see apic gateway-extensions.

Note: You cannot upload more than one Gateway extension .zip file to the same Gateway service. If you want add further extensions later, update the original .zip file, then use the `apic gateway-extensions:update` command to replace the previous gateway extensions file with the revised one; for example:

```
apic gateway-extensions:update mynewextension.zip --scope org --org admin --gateway-service mygatewayservice --
availability-zone myavailabilityzone --server mgt-admin.myserver.com
```

3. Apply the extension to each server by restarting the API Connect gateway service object; complete the following steps on **each** Gateway server in the Gateway service:
Note: In Kubernetes environments, the following steps can be accomplished by issuing `kubectl delete pod` on **each** gateway pod. Be sure to wait until each pod gets back to the ready state before deleting the next gateway pod.
    a. Remove the Gateway server from the load balancing group.
    b. Disable, then enable the API Connect gateway service object. You can do this by using either the DataPower administrative user interface in a web browser, or by using the DataPower CLI.
        - By using the administrative user interface:
            - Log in to the DataPower administrative user interface; for the Graphical Interface, select WebGUI rather than Blueprint Console.
            - Switch to the DataPower application domain for API Connect if necessary.
            - Search for `API Connect Gateway Service`.
            - Set the Administrative State to disabled.
            - Apply the changes.
            - Set the Administrative State to enabled.
            - Apply the changes.
        - By using the CLI:
            - Enter configuration mode by entering the command `configure`.
            - Navigate to the DataPower application domain for API Connect by entering the command `switch api_connect_domain_name`, where *api_connect_domain_name* is the name of your API Connect application domain.
            - Disable the API Connect gateway service object by entering the following command:

              ```
              apic-gw-service; admin-state disabled; exit
              ```

            - Enable the API Connect gateway service object by entering the following command:

              ```
              apic-gw-service; admin-state enabled; exit
              ```

    c. Ensure that the gateway service object initialization has completed.
    d. Re-add the Gateway server to the load balancing group.
4. Apply the extension to each server by restarting the API Connect gateway service object; complete the following steps on **each** Gateway server in the Gateway service:
Note: In Kubernetes environments, the following steps can be accomplished by issuing `kubectl delete pod` on **each** gateway pod. Be sure to wait until each pod gets back to the ready state before deleting the next gateway pod.

Results

The extension is uploaded and applied to each of the servers in the Gateway service, and the associated enforcement capabilities are applied to all incoming API resource requests.

Warning: DataPower Gateway (v5 compatible) only If you upload an extension to the DataPower Gateway (v5 compatible), a status file called extension_import_response.xml is written to the local:/ext folder in the gateway file system. Do not remove this file, otherwise if your gateway extension is subsequently removed then the referenced objects cannot be reverted to their original state automatically, and you will therefore have to either complete these clean up tasks manually, or re-add the gateway extension to regenerate the extension_import_response.xml file and then remove the gateway extension again. If the extension_import_response.xml exists and there is a cleanup failure for any object, a failure file is written to the local:/ext folder, with the failure details.

Cloud Manager Tutorials

Tutorials for using the Cloud Manager user interface in IBM® API Connect.

Prerequisites

To complete the following tutorials you must be the Cloud Administrator or Cloud Owner of an API Connect instance.

Overview

These tutorials guide you through common tasks performed to set up and maintain the configuration of a cloud.

- **Tutorial: Configuring the Cloud**
  This tutorial shows you how to create a basic cloud configuration, with available gateway, analytics and developer portal services.
- **Tutorial: Creating a Provider Organization**
  This tutorial shows you how to create a Provider Organization.

Tutorial: Configuring the Cloud

This tutorial shows you how to create a basic cloud configuration, with available gateway, analytics and developer portal services.

Before You Begin

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator

You will need the following information to complete this tutorial.

- The IP address or FQDN of the gateway, and the port assigned to accept API requests from clients.
- The IP address or FQDN of the gateway, and the port assigned to communicate with the API Management server. This cannot be the same as the port for API requests.
- The FQDN of the Developer Portal service to accept requests from clients.
- The FQDN of the Developer Portal to communicate with the API Management server.
- The FQDN of the Analytics service to accept requests from clients.
- The address, port and optional login credentials for an SMTP email server.

Note: The port for the **`portal-www`**, **`portal-admin`**, and **`analytics-ingestion`** services is **`443`**, and isn't configurable.

About this tutorial

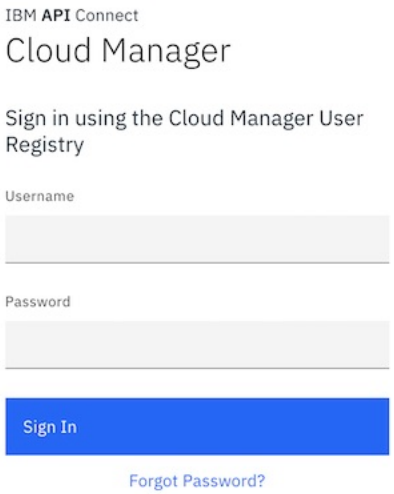
In this tutorial you are going to complete the following lessons:

- Initial Cloud Manager Console login
- Configure an Email Server
- Register a Gateway Service
- Register a Portal Service
- Register an Analytics Service
- Configure a Default Gateway Service

Initial Cloud Manager Console login

Take the following steps to log in to the Cloud Manager user interface for the first time.

1. In a web browser, enter the management service URL. For example, `https://ManagementService.domain/admin` where ManagementService.domain is the fully qualified host name or IP address of the Management service.



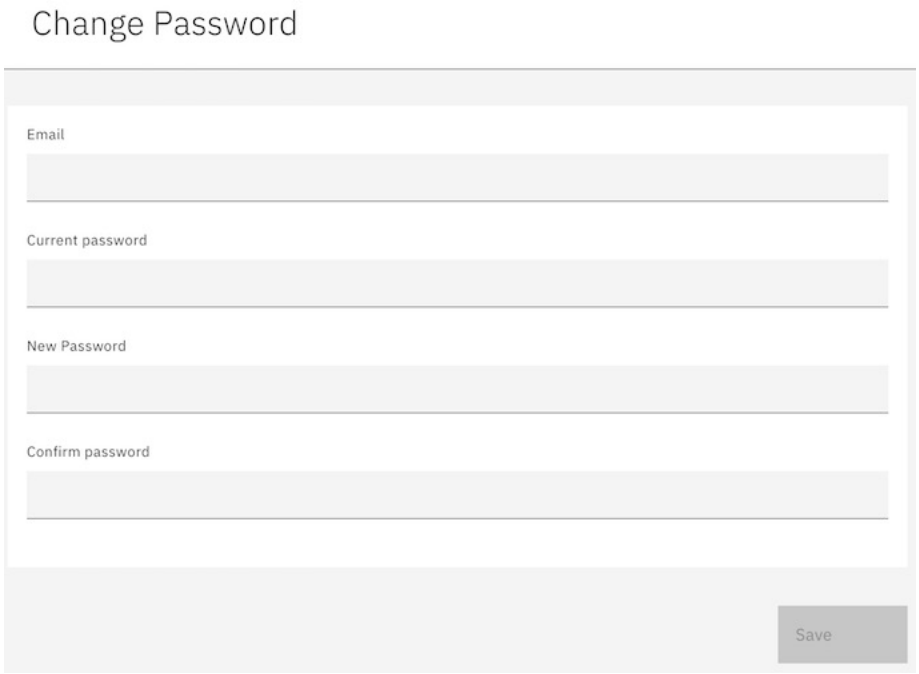
2. Enter the Cloud Administrator user name and password. The default values are `admin` for the user name and `7iron-hide` for the password.

   OpenShift top-level CR install

   If you installed API Connect on OpenShift using the top-level "API Connect Cluster" CR, the default user name is `admin`. If you are logging in for the first time, run the following commands to retrieve the initial password, which is generated automatically during deployment:

   ```
   oc get secret -n <namespace> | grep mgmt-admin-pass
   oc get secret -n <namespace> <secret_name_from_previous command> -o jsonpath="{.data.password}" | base64 -d && echo ""
   ```

3. You are immediately required to change the admin password, as well as provide an email address for the cloud administrator. Enter the necessary information. Note: If you forget your password and request a password reset, the notification email is sent to this email address. This action will use the email server set in the Notifications section of the cloud Settings. This tutorial shows you how to set this configuration.

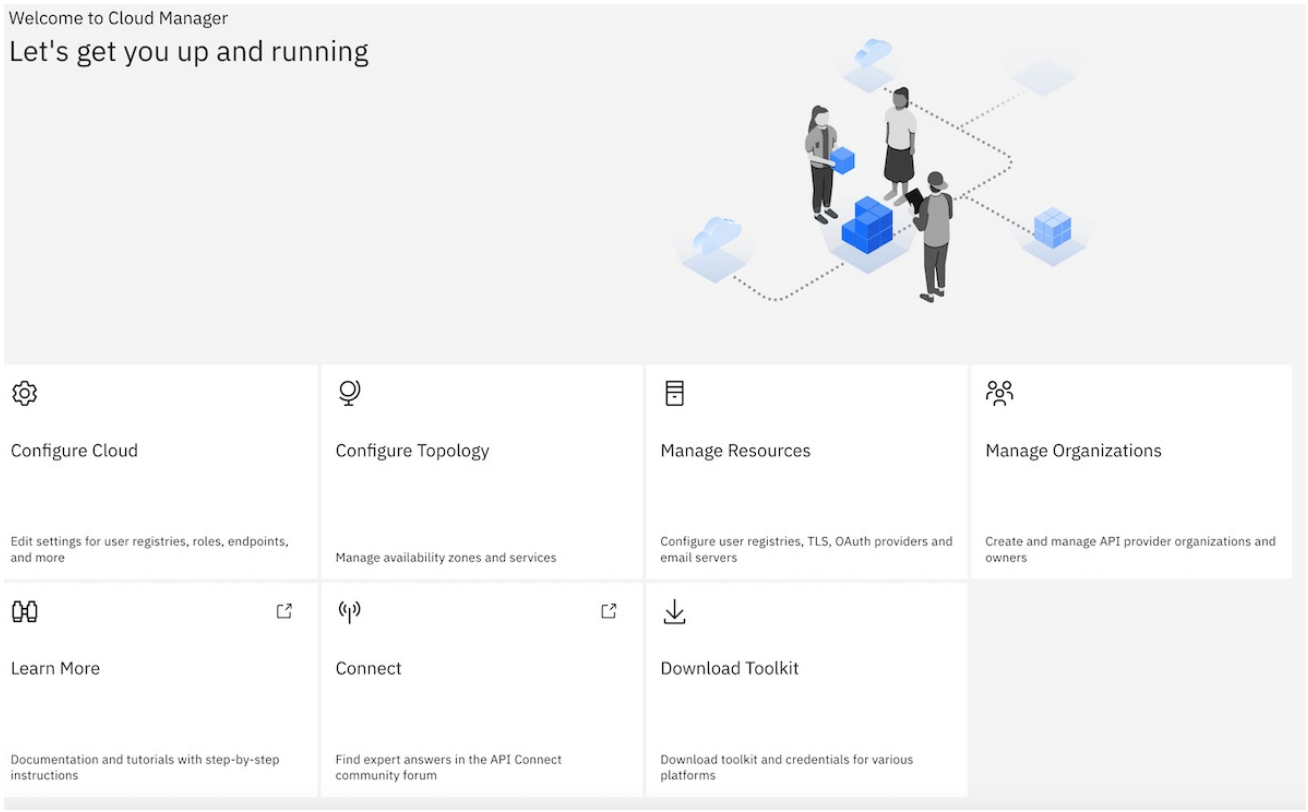


4. Click Save.
5. Log in using the new password.

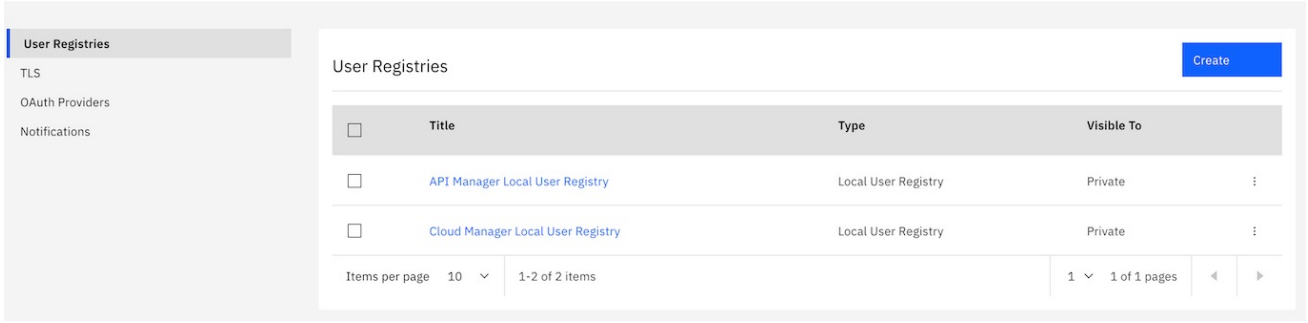
Configure an Email Server

Take the following steps to configure an email server. This configuration allows you to recover the admin password or receive other important notifications.

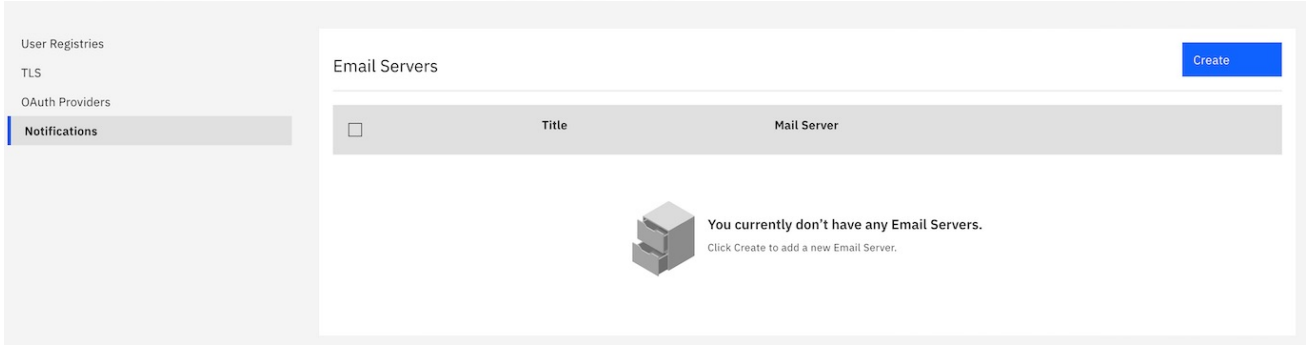
1. Click Resources icon or the Manage Resources tile.



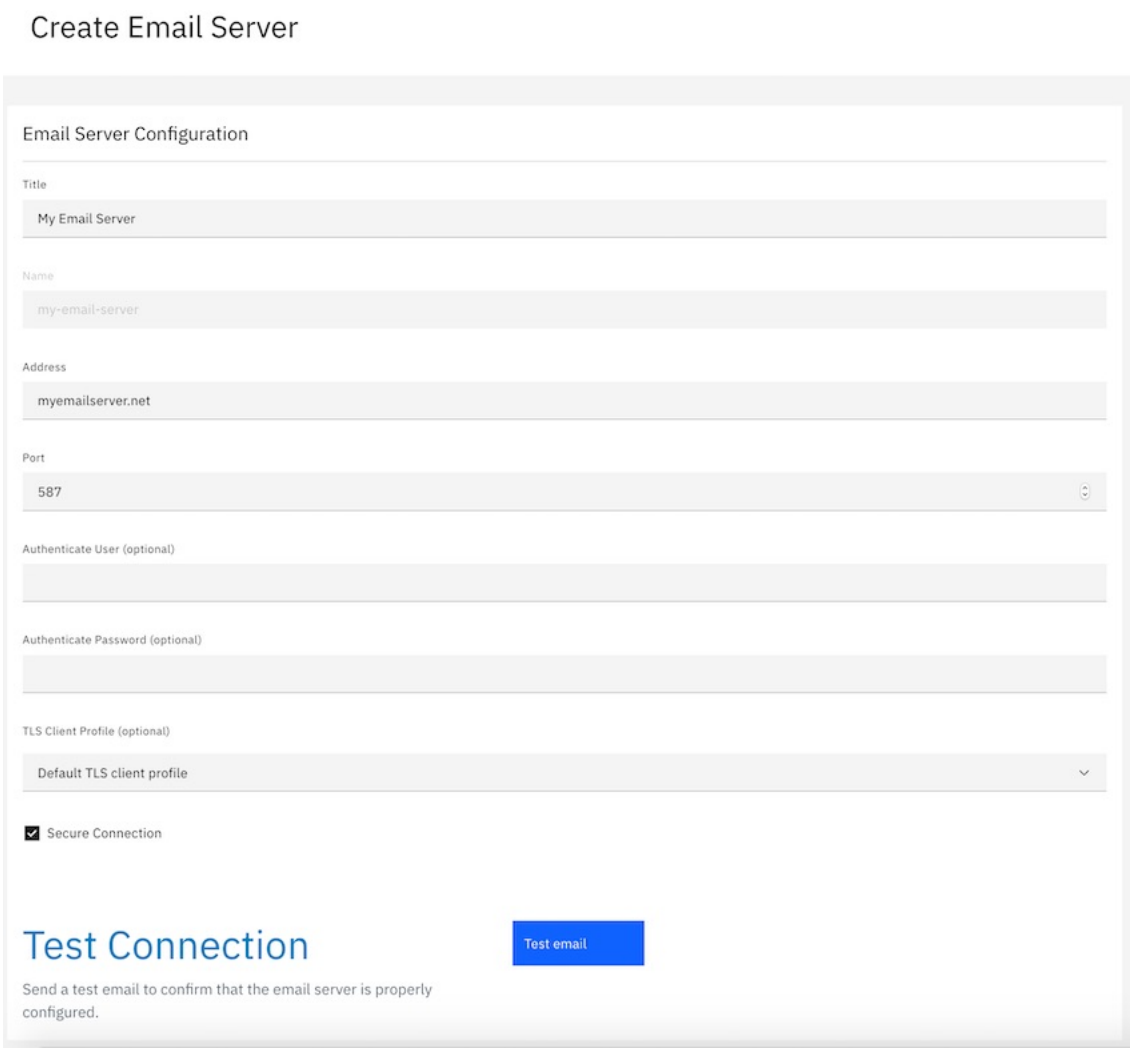
2. Click Notifications.



3. Click Create.



4. Enter the appropriate values in the fields These values vary depending on your site.
5. Click Save.

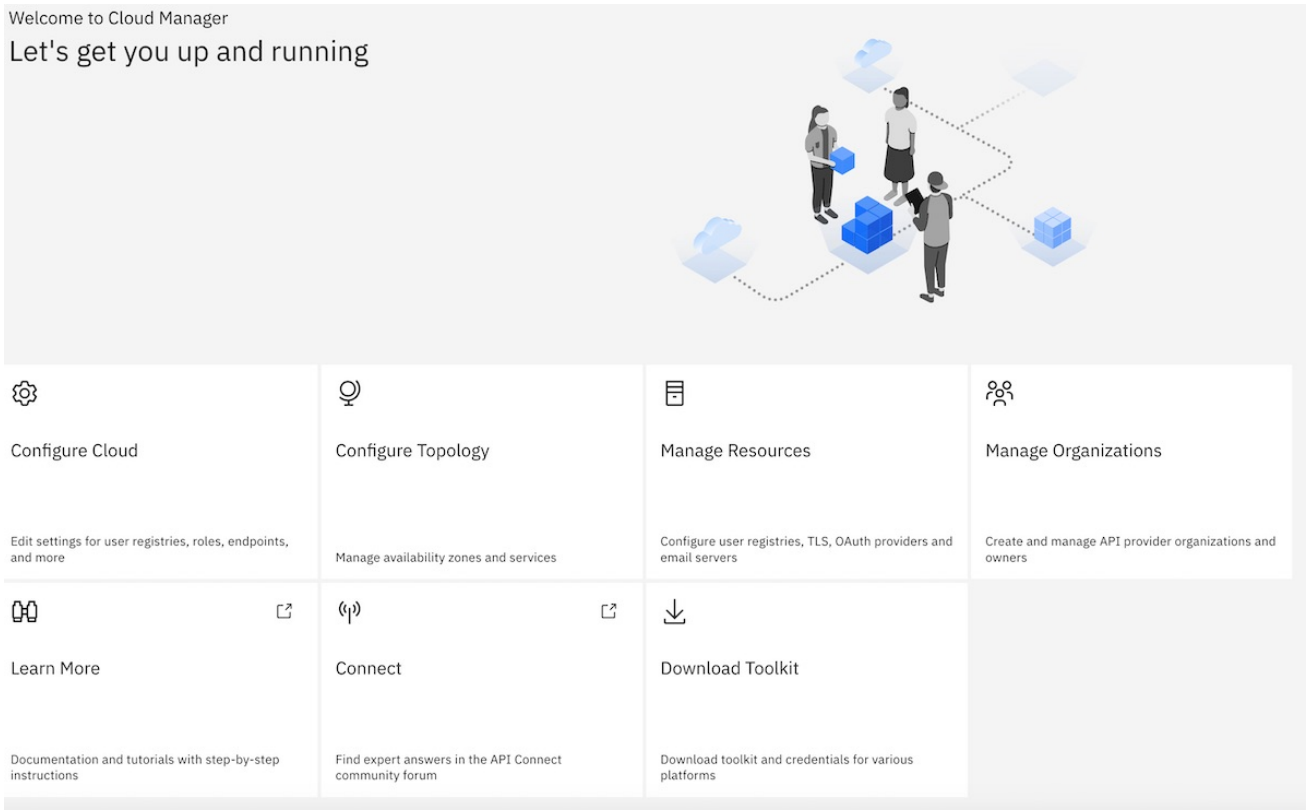


6. Click the Settings icon, then click Notifications > Edit.
7. Ensure your email server is selected, then click Save.
8. Click the Home icon.

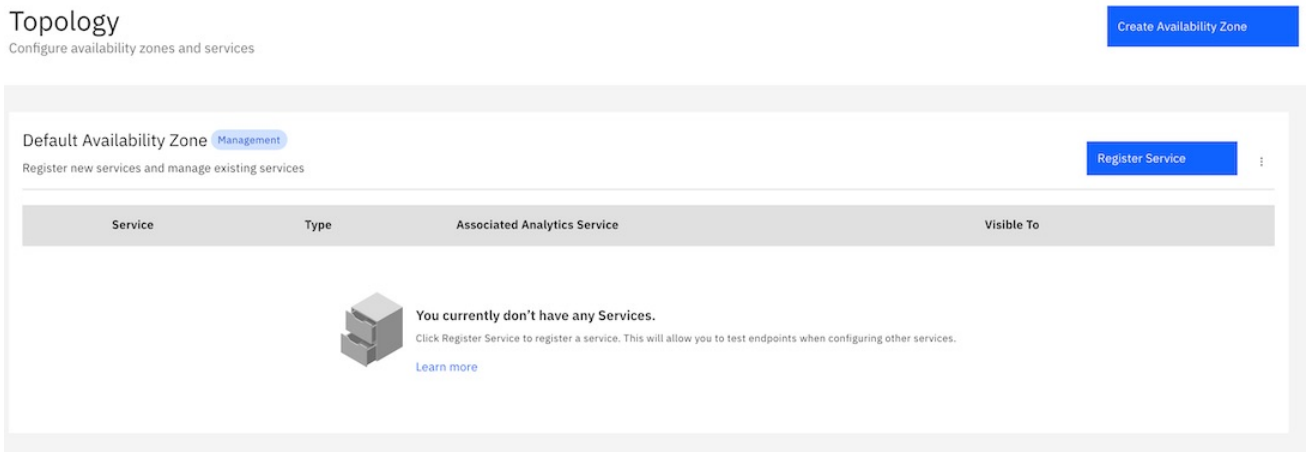
Register a Gateway Service

Take the following steps to register a gateway service.

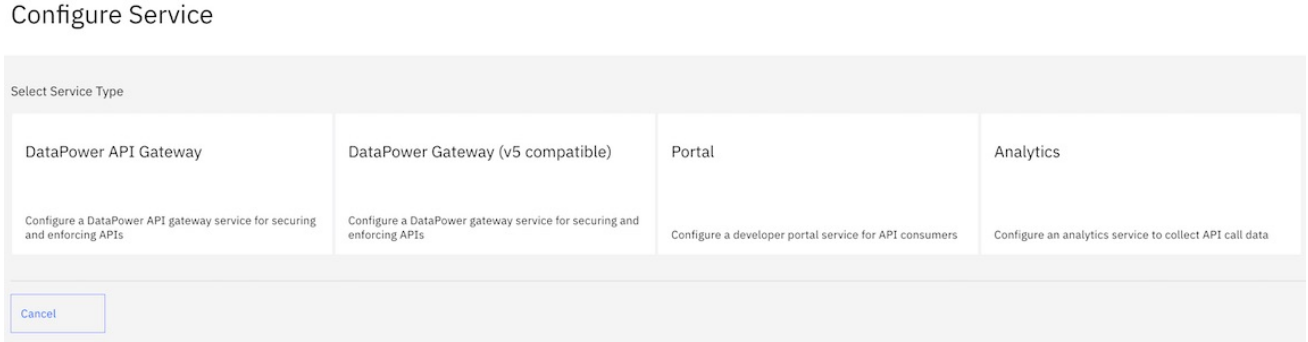
1. Click the Configure Topology tile.



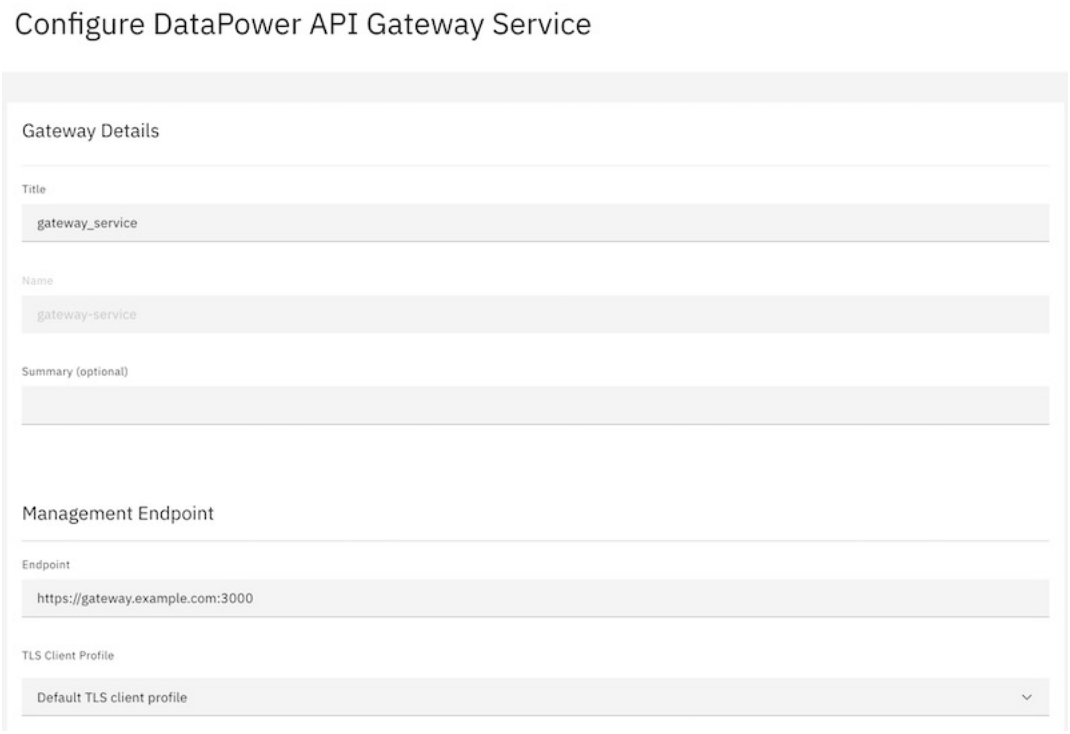
2. Click Register Service.



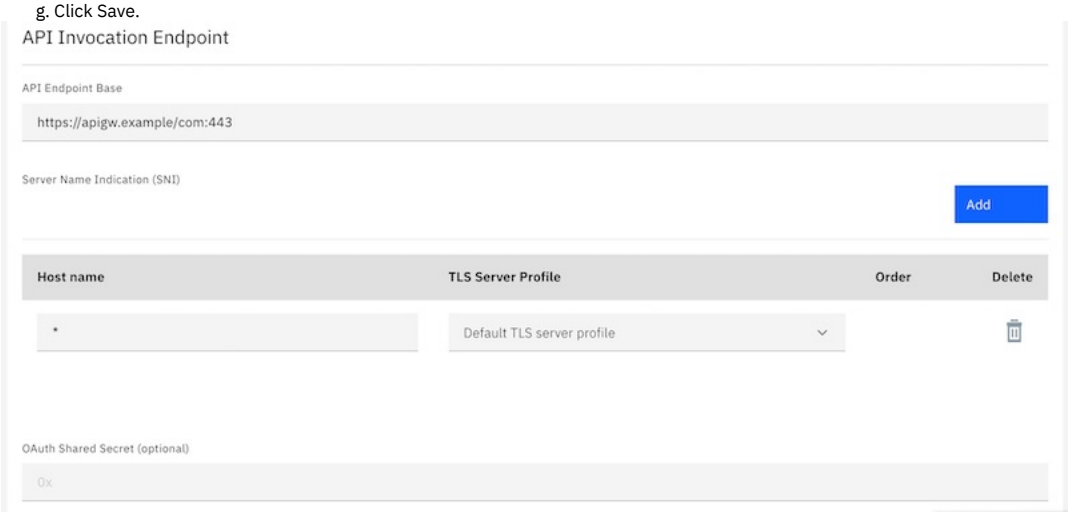
3. Click DataPower API Gateway.



4. Take the following steps.
   a. Enter `gateway_service` in the Title field.
   b. In the Management Endpoint section, enter the URL of the address and port assigned to the management endpoint in the Endpoint field. This is the port used by the API Management server to connect to the gateway.
   c. Leave the remaining values as given to set TLS profiles.



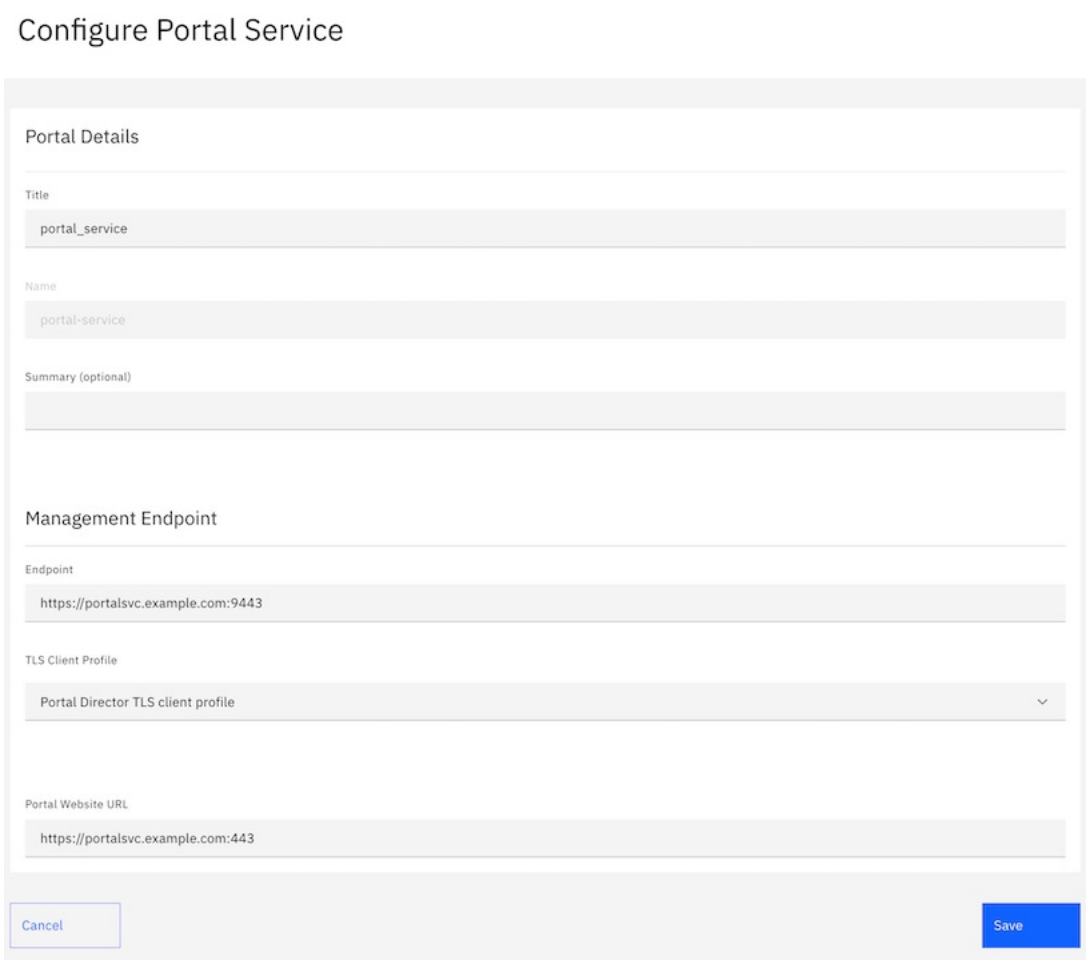
d. In the API Invocation Endpoint section, enter the URL of the gateway address and port assigned to accept API requests from clients in the API Endpoint Base field.
e. Do not change the defaults in the Server Name Indication (SNI) fields. Note that your topology may require specific values in these fields.
f. Optionally enter a hex value in the OAuth Shared Secret field. This must be a 64-bit hex value that begins with "0x". Providing a shared secret value here enables gateways in a cluster to read the OAuth tokens generated by any member of the cluster using the same secret.
g. Click Save.



Register a Portal Service

Take the following steps to register a portal service.

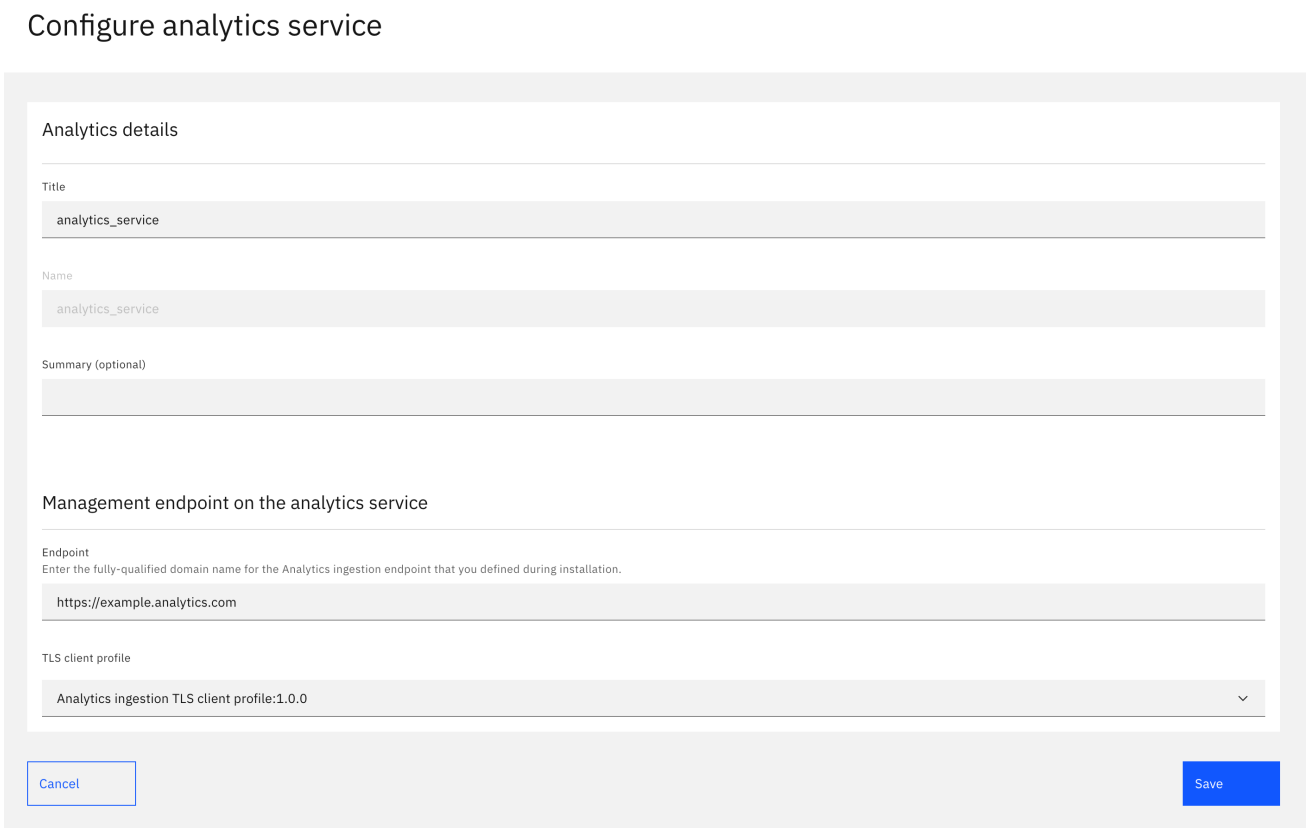
1. Click Register Service.
2. Click Portal.
3. Take the following steps.
   a. Enter `portal_service` in the Title field.
   b. In the Management Endpoint section, enter the URL of the portal address and port assigned to communicate with the management server in the Endpoint field.
   c. Enter the URL of the portal address and port assigned to accept requests from clients in the Portal Website URL field.
   d. Use the reconfigured profile in the TLS Client Profile field.
   e. Click Save.



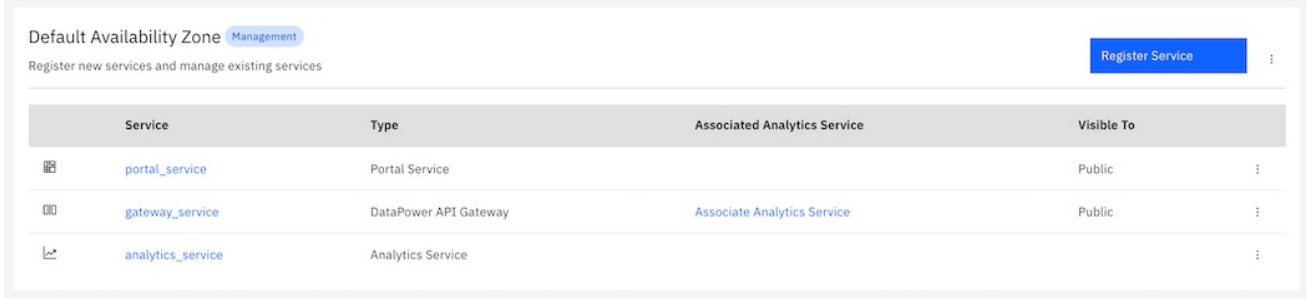
Register an Analytics Service

Take the following steps to register an analytics service.

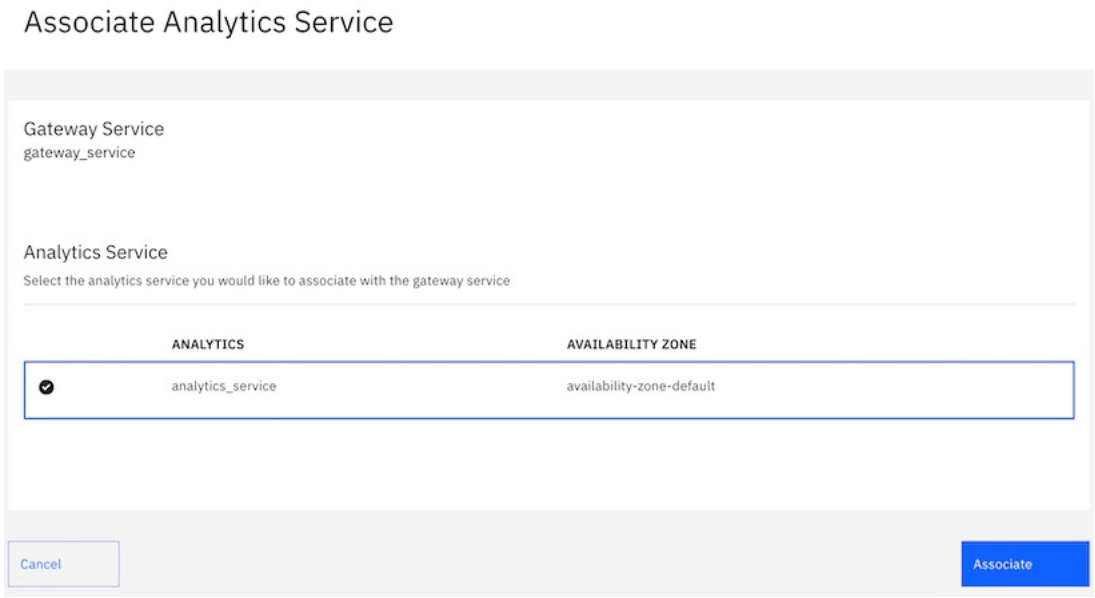
1. Click Register Service
2. Click Analytics.
3. Take the following steps.
   a. Enter `analytics_service` in the Title field.
   b. In the Management Endpoint section, enter the URL of the analytics server address and port assigned to accept requests from clients in the Endpoint field.
   c. Select `Analytics ingestion TLS client profile` in the TLS Client Profile field.
   d. Click Save.



4. Click Associate Analytics Service corresponding to the gateway service listing.



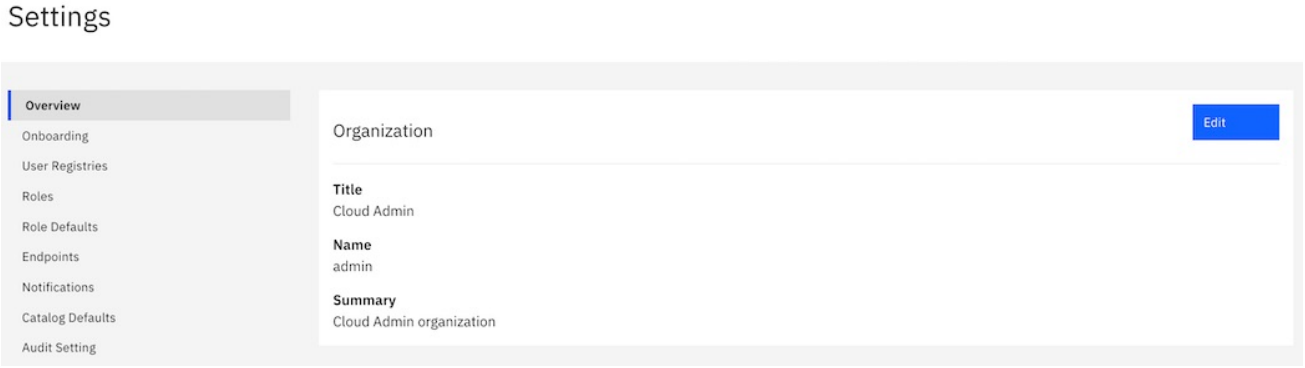
5. Select analytics_service. Click Associate.



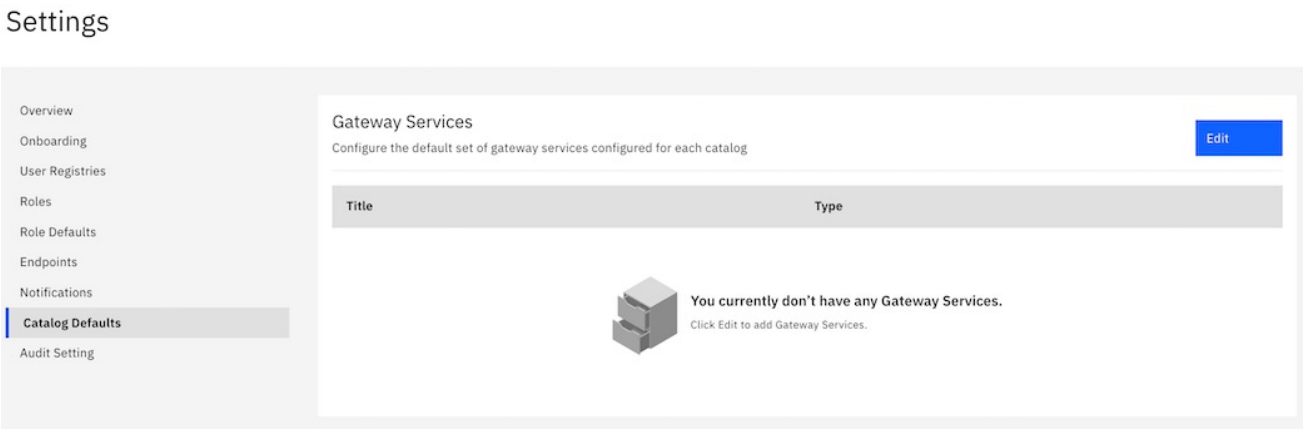
Configure a Default Gateway Service

Take the following steps to configure a default gateway service for every catalog in the cloud.

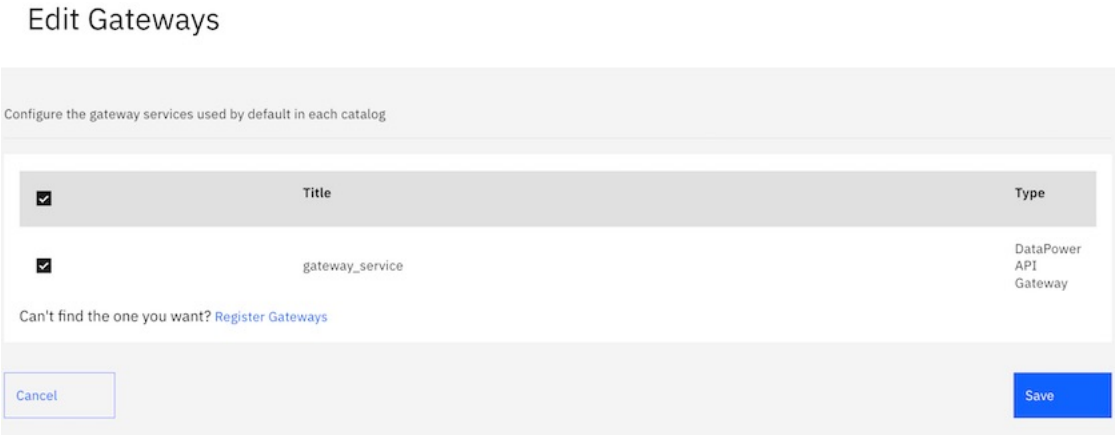
1. Click Settings icon .
2. Click Catalog Defaults.



3. Click Edit.



4. Select an available gateway service.
5. Click Save.



What you did in this tutorial

- Set new Cloud Manager Console login password
- Configured a Gateway Service
- Configured a Developer Portal Service
- Configured an Analytics Service
- Configured a Notifications Email Server
- Configured a Catalog Default Gateway Service

Related information

- All tutorials

Tutorial: Creating a Provider Organization

This tutorial shows you how to create a Provider Organization.

Before You Begin

This task can be completed by users who are assigned one of the following roles:

- Cloud Owner
- Cloud Administrator

You must also complete the following tasks before beginning:

- Configuring an email server for notifications
- Setting up notifications

Note that an email server must be configured and an active email server must be selected before a provider organization account can be created.

About this tutorial

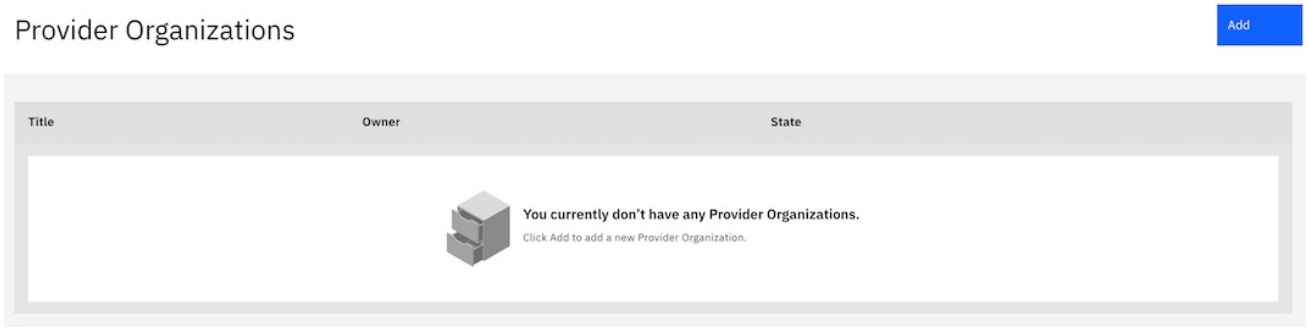
In this tutorial you are going to complete the following lessons:

- Create a Provider Organization

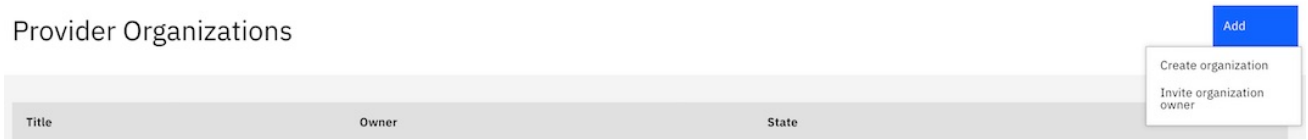
Create a Provider Organization

Take the following steps to create a new Provider Organization.

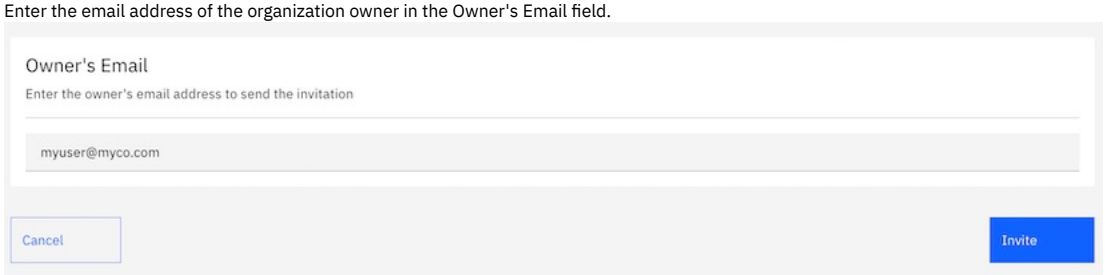
1. Log in to the Cloud Manager.
2. Click Provider Organizations.



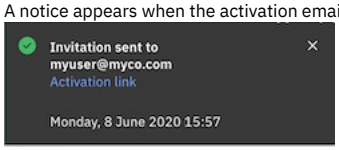
3. Click Add > Invite organization owner.



4. Enter the email address of the organization owner in the Owner's Email field.

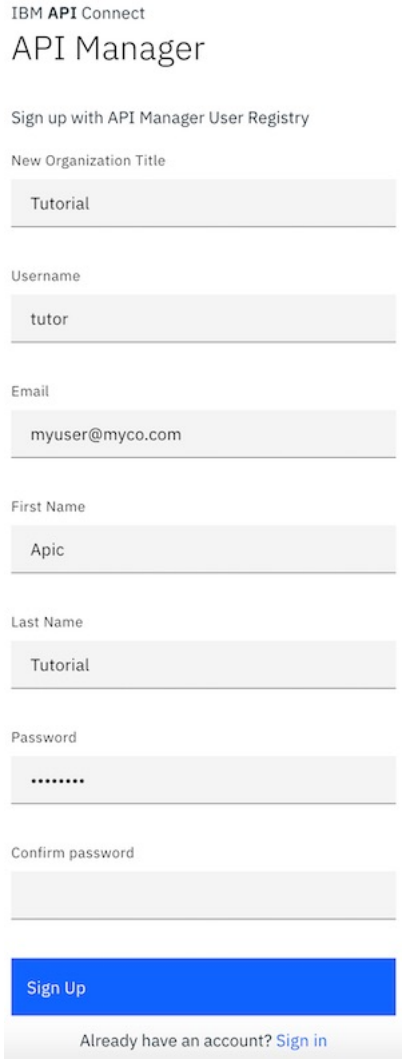


5. Click Invite.
6. A notice appears when the activation email has been sent. Click Activation link..

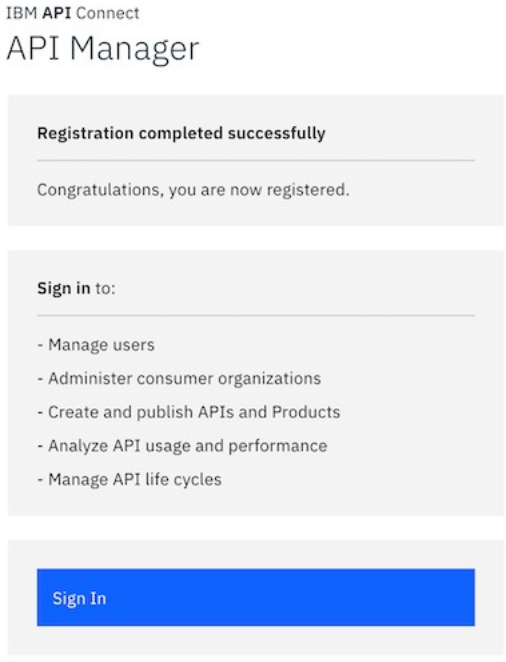


7. A dialog box displays the activation link. Click the Copy to clipboard icon .

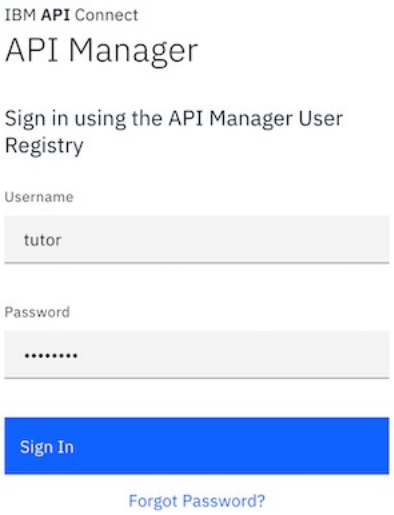
8. Open a new window in your browser. Paste the copied activation link into the Location bar. Optionally, you can click the activation link in the invitation email to open the registration form.
9. Complete fields shown in the form to sign up with the API Manager User Registry. Click Sign up.



10. You see a confirmation of registration. Click Sign in.



11. Enter the username and password you just created. Click Sign in to begin creating APIs and products.



12. Return to the original window of your browser. Click Provider Organizations to refresh the list. Your new organization is listed.

What you did in this tutorial

In this tutorial, you completed the following activities:

- Created a new Provider Organization.

Related information

- Importing an API

Developing your APIs and applications

You develop APIs and LoopBack® applications by using the IBM® API Connect developer toolkit.

The API Connect developer toolkit provides both the API Designer user interface and a command line interface that you can use to develop APIs and LoopBack applications and publish them to API Connect.

You publish APIs by including them in a Product and then publishing the Product. You define your APIs and Products by creating and validating YAML definition files in your local file system. You can then interact with API Connect by using either the API Designer or the toolkit commands.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
The developer toolkit is described in detail in the following subsections:

- **API development checklist**
  A summary, with links, of the key tasks for developing APIs in IBM API Connect.
- **Working with the toolkit**
  Install the developer toolkit by downloading an executable file. You can then use the toolkit commands to interact with IBM API Connect and publish APIs that you have defined in your file system.
- **Managing platform REST API keys**
  Create API keys associated with your user account, and then use them to authenticate when you call the platform REST APIs provided by API Connect.
- **Searching for items in API Manager**
  Use the search feature in IBM API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.
- **Working with API definitions**
  An API definition specifies the complete configuration for an API. You can create and configure your APIs either by using either the API Designer UI application, or by using the browser based API Manager UI.
- **Working with Products**
  In IBM API Connect, Plans and APIs are grouped together in Products, with which you can manage the availability and visibility of APIs and Plans.

- **Creating and validating API and Product definitions by using the CLI**
  The developer toolkit of IBM API Connect provides a command line interface that you can use to create and publish API and Product definitions, and also to validate YAML or JSON definitions.
- **Working with global policies**
  Use global policies to configure policy assemblies that are called just before, or just after, each of your API assemblies is called, or when an error is thrown. You can upload global policies into each of the gateway services in your Catalogs, and then designate, for each gateway service, which global policy should be called before an API assembly is called, after an API assembly is called, or if an error occurs when an API assembly is called. The designated global policies are applied to all the APIs that are deployed to the associated gateway service.
- **Reference**
  Reference information for developing your APIs in API Connect, including context variables, and template variables.

API development checklist

A summary, with links, of the key tasks for developing APIs in IBM® API Connect.

Task	Description
Activate your API Manager user account	For you to be able to connect to a Management server by using either the API Designer or API Manager user interfaces, one or other of the following prerequisites must be satisfied: • Your administrator has created a provider organization, specifying you as the owner; see Creating a provider organization. • You have been added as a member of a provider organization; see Adding provider organization users and assigning roles.
Install the developer toolkit	The developer toolkit provides a command-line environment and an API Designer user interface, both of which enable you to develop and work with your APIs on your local machine; you publish your APIs to the Management server when ready. You can also develop your APIs directly on the Management server by using the API Manager user interface.
Log in to the user interface; see Logging in from API Designer, or Accessing the API Manager user interface, depending on which user interface you want to use.	After you have activated your account, you can log in to your Management server from either user interface. You can also use the API Designer in offline mode, but to be able to test and publish APIs you must be connected to a Management server.
Create your API definitions	You can create API definitions in the following ways: • Create a REST API definition either by composing the API definition, and its operations, from scratch, creating a proxy API that calls an existing endpoint, or importing an OpenAPI definition. • Expose an existing SOAP service as an API. • Create a GraphQL API proxy definition that proxies a backend GraphQL server.
• Secure your APIs (OpenAPI 2.0) • Secure your APIs (OpenAPI 3.0)	You configure security for an API by creating one or more security definitions that you then apply to your API, and to the operations in your API. Security mechanisms include basic authentication through a user registry, and token based authentication with OAuth.
Test your APIs	The user interfaces provide a test tool with which you can test your APIs through the gateway.
Productize and publish your APIs	To make your APIs available to application developers, you include them in a Product that you then publish to the gateway.
Apply visibility controls and rate limiting to your APIs	Visibility settings control which APIs the application developers can subscribe to. Rate limiting controls the usage of your APIs; for example, you can apply a rate limit of 100 calls per minute.
Develop your APIs by using the developer toolkit CLI	You can use the developer toolkit CLI to develop APIs based on OpenAPI definition files in YAML or JSON format.

Working with the toolkit

Install the developer toolkit by downloading an executable file. You can then use the toolkit commands to interact with IBM® API Connect and publish APIs that you have defined in your file system.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
The following subsections describe how to install and use the toolkit:

- **Installing the toolkit**
  You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM API Connect.
- **Logging in from API Designer**
  To log in from the API Designer user interface for IBM API Connect, you will need the host name of the management server and the user name and password associated with your API Manager account.
- **Using the developer toolkit command-line tool**
  The IBM API Connect developer toolkit provides a command-line tool, `apic`, that you can use to perform all API Connect tasks. You can also use the command-line tool to script tasks such as continuous integration and delivery.

Installing the toolkit

You can install the toolkit that provides CLI commands, and the API Designer user interface, for IBM® API Connect.

About this task

The toolkit is provided as executable files, so no actual installation is necessary, you just to need to download the required compressed file and extract the contents.

There are two toolkit options available:

- CLI: provides a command line environment for working with IBM API Connect.
- CLI + LoopBack + Designer: provides a command line environment for working with IBM API Connect, including LoopBack® support, and the API Designer application.

To install the toolkit, download the compressed file that is appropriate for your chosen toolkit option and platform, then extract the contents to a chosen location on your local machine. The compressed file contains an executable file for running CLI commands and, if you choose the CLI + LoopBack + Designer option, an executable file for launching the API Designer application.

You can download the toolkit from one of the following locations:

- From IBM Fix Central.
- From the Cloud Manager or API Manager user interface.

Procedure

To install and run the toolkit, complete the following steps:

1. Download the toolkit compressed file.
   - To download the toolkit from IBM Fix Central, complete the following steps:
     - Open the IBM Fix Central site in your browser.
     - In the Product selector field, enter `API Connect`, then select IBM API Connect from the drop down list.
     - Select your Installed Version and click Continue. If you do not know your installed IBM API Connect version, contact your administrator.
     - In the Text field, enter `toolkit`, then click Continue.
     - Select the toolkit file that you want to download.
     - Click Continue, then follow the instructions to complete the download operation.
   - To download the toolkit from Cloud Manager or API Manager user interface, complete the following steps:
     - Open the Cloud Manager or API Manager user interface.
     - On the welcome page, click the Download Toolkit tile. The two toolkit options are listed.
     - Click your platform alongside either the CLI option or the CLI + LoopBack + Designer option as required, then save the associated compressed file to your local file system.
     - To download the API Designer credentials (Client ID and Client Secret), click Download alongside API Designer Credentials, then save the designer_credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 6.
     - To download the toolkit credentials (Client ID and Client Secret), click Download alongside Credential, then save the credentials.json file to your local file system. A command is provided for installing the credentials, as described in detail in step 5.
     - Close the Install API Connect Toolkit window.
2. Extract the contents of the toolkit compressed file to a folder of your choice.
   The contents of the file depend on the your chosen toolkit option and platform, as follows:

   - The apic-slim or apic-slim.exe file is the CLI for IBM API Connect.
   - The apic or apic.exe file is the CLI for IBM API Connect including LoopBack support.
     Tip: If you are using the CLI option, then if you rename the apic-slim file to apic, or the apic-slim.exe file to apic.exe, you can run the CLI commands exactly as documented, copy and paste sample commands from the documentation, and use any command scripts as-is if you later move to the CLI + LoopBack + Designer option.
   - The api_designer-*platform* file is the API Designer user interface application for the specified platform.
3. Optional: Delete the **`$HOME/.apiconnect/node_clis`** directory.
   You need to do this only if you replaced a version of **`apic`** or apic.exe and are using the **`apic lb4`** command. You need to delete the directory get **`apic`** to unpack the new loopback.

   On Windows, $HOME is defined by environment USERPROFILE.

4. Run the CLI.
   - For the Mac OSX or Linux® platforms, complete the following steps:
     - Open a terminal instance and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Make the CLI file an executable file by entering the following command:

       ```
       chmod +x download_name
       ```

       Where *download_name* is the name of the toolkit file that you downloaded, either apic or apic-slim.
     - Run CLI commands as follows:

       ```
       ./apic command_name_and_parameters
       ```

       or

       ```
       ./apic-slim command_name_and_parameters
       ```

       For details of the CLI commands, see apic.
   - For the Windows platform, complete the following steps:
     - Open a command prompt and navigate to the folder where you extracted the contents of the toolkit compressed file.
     - Run CLI commands as follows:

       ```
       apic command_name_and_parameters
       ```

       or

       ```
       apic-slim command_name_and_parameters
       ```

For details of the CLI commands, see apic.

Tip: Add the folder location of your CLI file to your PATH variable so that you can run CLI commands from anywhere in your file system.

5. Install the toolkit credentials.
   If you do not install the toolkit credentials that are provided for download, as detailed previously in the toolkit credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the credentials into your local toolkit, run the following command:

   ```
   apic client-creds:set toolkit_credentials_file_path/credentials.json
   ```

   where *credentials_file_path* is the location to which you downloaded the toolkit credentials JSON file. After you have run this command, your toolkit uses these new credentials to authenticate with the management server.
   Note: At any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.
   To revert to the default toolkit credentials for all login operations from the toolkit CLI, use the following command:

   ```
   apic client-creds:clear
   ```

   For increased security, an administrator can remove the default credentials from the management server by completing the following steps:
   a. Log in to the management server as an administrator; see Logging in to a management server.
   b. Delete the default credentials by running the following commands:

   ```
   apic registrations:delete toolkit --server mgmt_endpoint_url
   apic registrations:delete consumer-toolkit --server mgmt_endpoint_url
   ```

6. Install the API Designer credentials.
   If you do not install the API Designer credentials that are provided for download, as detailed previously in the API Designer credentials download instructions, API Connect uses a default set of credentials that are identical for all deployments. However, the downloaded credentials were generated during the deployment of your API Connect system and are unique to your installation. To install the custom credentials into your local API Designer, set the **APIC_DESIGNER_CREDENTIALS** environment variable to the credentials download location, using the mechanism appropriate for your operating system.

   After you have set the **APIC_DESIGNER_CREDENTIALS** environment variable, your API Designer uses the new credentials to authenticate with the management server.

   - Windows:
     If you are using Windows, create an environment variable using one of the following methods:

     - Create a permanent environment variable so that you can start API Designer from any location on your computer:
       - Open the Environment Variables page: Click Start > Settings > System and in the "Related Settings" section of the page, click Advanced System Settings.
       - On the Advanced tab of the System Properties dialog box, click Environment Variables.
       - In the "User variables" section, click New and create an environment variables with the following settings:
         - Variable: **APIC_DESIGNER_CREDENTIALS**
         - Value: *<designer_credentials_file_path>*\designer_credentials.json where ***<designer_credentials_file_path>*** is the location where you stored the designer_credentials.json file.
       - Click OK to save the new environment variable, and then exit the dialog box.
     - If you don't have the required permissions to create a permanent environment variable, you can create a temporary one that will only be used while you are running the API Designer application. The following steps must be performed every time you start API Designer:
       - Open the Windows command prompt.
       - Run the following command to set the temporary environment variable:

         ```
         set APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>\designer_credentials.json
         ```

         where ***<designer_credentials_file_path>*** is the location where you stored the designer_credentials.json file.

         Note: Leave the command prompt open for the next step. You must set the temporary variable and start the API Designer within the same Windows command session.
       - Now run the following command to start API Designer:

         ```
         C:\"Program Files\API Designer\API Designer.exe"
         ```

         By default, API Designer is installed in the C:\"Program Files\API Designer folder as shown in the example. If you installed it into a different location, use your own location in the command. Note that the path and file name are enclosed in quotation marks because they contain spaces.

   - Mac:
     If you are using Mac OS X, create an environment variable using one of the following methods:

     - Run the following command to set the global environment variable:

       ```
       launchctl setenv APIC_DESIGNER_CREDENTIALS <designer_credentials_file_path>/designer_credentials.json
       ```

       where ***<designer_credentials_file_path>*** is the location where you stored the API Designer credentials JSON file.
     - Pass in the environment variable while starting API Designer from the command line. With this method, you must run the following command every time you start API Designer:

       ```
       APIC_DESIGNER_CREDENTIALS=<designer_credentials_file_path>/designer_credentials.json open
       <designer_application_file_path>/'API Designer.app'
       ```

       where:
       - ***<designer_credentials_file_path>*** is the location to which you downloaded the API Designer credentials JSON file.
       - ***<designer_application_file_path>*** is the location to which you downloaded and uncompressed the API Designer application.

7. Start the API Designer application from the location to where you stored the extracted file.
   Note:
   - To uninstall the API Designer application on a Windows platform with a non Administrator account, complete the following steps:
     - In Windows File Explorer, navigate to the USER_HOME\AppData\Local\Programs\api-designer folder.
     - Run the **Uninstall API Designer application** application. Do **not** use the **Add or remove programs** window.

- To uninstall the API Designer application on a Windows platform with an Administrator account, you can either run the **Uninstall API Designer application** application, or you can use the **Add or remove programs** window.

Results

The IBM API Connect toolkit CLI and, if selected, the API Designer user interface application are installed on your local system.

For information on using the API Designer user interface, see Developing your APIs and applications.

For information on using the toolkit CLI, see Using the developer toolkit command-line tool.

Logging in from API Designer

To log in from the API Designer user interface for IBM® API Connect, you will need the host name of the management server and the user name and password associated with your API Manager account.

About this task

API Designer provides a secure login to a management server for staging and publishing APIs and Products. You log in to the same host that was configured for API Manager and use your API Manager user name and password. You will need an account on API Manager and membership in a provider organization to access the API Designer.

You can also use the Explorer tab to see how your APIs look to a consumer in the Developer Portal. You can check the descriptions of the different artifacts, and refine any schemas or examples, and you can also use the Try it tool to test the behavior of the API.

Note:

- Configuration changes that are made in API Manager are not available in API Designer concurrently. You will need to log out of API Designer and log back in to access any updates.
- Set **`SKIP_IBMID_LOGIN=true`** as an environment variable to bypass the IBM ID login screen for API Designer. This applies to users who are working on an internal network, without access to external networks.

Procedure

To log in to API Designer, complete the following steps:

1. Download and extract API Designer, as described in Installing the toolkit.
2. Start the API Designer user interface by running the application from the location where you extracted the contents of the toolkit compressed file.
   Note: If you see a message stating that credentials cannot be found, download new credentials as explained in Downloading the toolkit.
3. Optional: If you see the Accept License page, click the URL read the license agreement, then click the button to accept the license. This prompt appears only one time. It does not appear once you have accepted the license on the current system.
4. Optional: If you see the IBM ID login page, enter your IBM customer ID.
   Note: You won't receive the IBM ID page if you are already authorized, or if you have set the **`SKIP_IBMID_LOGIN`** environment variable to **`true`**.
5. Open a directory on your local file system to store the API and Product specifications that you create in API Designer. The system defaults to a previously-accessed directory.
   If you want to switch to a different directory after logging in, click Switch directory on the menu bar in the header.

   Note: If you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, don't copy the file into your API Designer directory.
   Such an API will **not** be visible in the API Designer user interface if it doesn't contain the API Connect specific sections that are required by API Designer.

   Instead, copy the OpenAPI definition file to another folder, then import it into API Designer; the import operation adds the necessary API Connect specific sections; see Adding a REST API by importing an OpenAPI definition file.

   For more information on the API Connect specific sections in an OpenAPI definition file, see IBM extensions to the OpenAPI specification.

6. If this is the first time you have logged in, establish a connection to API Connect by completing the following steps:
   - Contact your administrator and request the "management endpoint URL" for connecting to API Connect.
     The management endpoint URL looks similar to the following example: https://*<host.domain>* and indicates the location where the API Manager is running.

     Note: If you are using API Connect as a component of Cloud Pak for Integration, the management endpoint URL follows this format:
     https://*cpd-your-company.domain*/integration/apis/*APIC_namespace*/*APIC_instance*

     Example: https://cpd-company.com/integration/apis/apic/production

     Do not add /manager to the URL. If you include /manager at the end of this URL, the connection with Cloud Pak for Integration will fail with a message stating that the host URL is invalid.

   - In the dialog window, enter the management endpoint URL as the Host URL and click Next.
   - Type the user name and password for your API Manager account.
   - Select a User Registry from the list.
   - Click Sign In.
7. For subsequent log ins, select the appropriate cloud where you will be working. You can select from any previous cloud connections.
8. When more than one cloud connection is configured, you can switch clouds when developing APIs and Products. See Switching clouds from API Designer.
   Note: If you are using an OIDC user registry that supports single sign-on (SSO), you cannot switch clouds to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then select the required cloud connection.
9. Click Add Another Cloud to add another cloud connection. Enter the URL, user name and password, and user registry to add the cloud connection.
   Note: If you are using an OIDC user registry that supports single sign-on (SSO), and you are already logged in, you cannot add a new cloud connection to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then add the new cloud

connection.
10. Delete a cloud connection by selecting the trash can icon.
11. API Designer will launch with the organization set to your provider organization in the selected cloud. If you have accounts for multiple provider organizations, it will default to the first organization in alphabetical order.
12. To choose another provider organization, select it from the drop-down list.

- **Setting the locale for API Designer**
  How to set one of the supported locales when you start the API Designer application.
- **Switching clouds from API Designer**
  You can switch clouds to work in another cloud in API Designer.
- **Working offline with API Designer**
  You can work offline (without a connection to the cloud) with API Designer to create APIs. Some functionality is missing due to the lack of a cloud connection.
- **Gathering support information in API Designer**
  When you report an issue with the API Designer application, enable debug mode and provide as much information as possible.

Related information

- Activating your API Manager user account
- Creating a provider organization account

Setting the locale for API Designer

How to set one of the supported locales when you start the API Designer application.

About this task

If you want to use a different locale than your current operating system (OS) locale, you can set a locale code by using the **`--lang`** argument when you start the API Designer application. This locale is then effective for the duration that the application is open. If no **`--lang`** argument is set, then the current OS locale is used.

Procedure

To change the locale for API Designer, complete the following instructions.

From the command line, run the following command to start the API Designer application with the **`--lang`** argument:

Mac OS X Mac:

```
open "API Designer.app" --args --lang=locale_code
```

Windows Windows:

```
"API Designer".exe --args --lang=locale_code
```

Where ***`locale_code`*** is one of the following supported locale codes:

Table 1. List of supported locale codes and their associated language

Locale code	Language
`cs`	Czech
`de`	German (Standard)
`en-US`	English (United States)
`es`	Spanish
`fr`	French (Standard)
`it`	Italian (Standard)
`ja`	Japanese
`ko`	Korean
`nl`	Dutch (Standard)
`pl`	Polish
`pt-br`	Portuguese (Brazil)
`ru`	Russian
`tr`	Turkish
`zh-cn`	Chinese (PRC)
`zh-tw`	Chinese (Taiwan)

For example, running the following command on a Mac:

```
open "API Designer.app" --args --lang=es
```

sets the API Designer language to Spanish.

Related tasks

- Installing the toolkit

Switching clouds from API Designer

You can switch clouds to work in another cloud in API Designer.

About this task

API Designer allows the flexibility to connect to multiple clouds. For example, your organization may have separate clouds for testing, development, and demos. You select the cloud connection when you log in, but you can also switch clouds when you are developing APIs and Products. Multiple clouds must be configured in order to switch clouds. You must have a valid API Manager account to connect to a cloud.

Note: If you are using an OIDC user registry that supports single sign-on (SSO), you cannot switch clouds to log in to the same user registry with a different user ID. You must first restart the API Designer application so that you are logged out of the registry, then select the required cloud connection.

Procedure

To switch clouds for working in API Designer, complete the following steps:

1. Log in to API Designer and select a cloud, or configure a new cloud connection.
2. From the Develop: APIs and Products screen, you can switch to another cloud by selecting it from the Switch Cloud Connection menu.
3. Delete a cloud connection by selecting the trash can icon.

Related information

- Activating your API Manager user account
- Creating a provider organization account

Working offline with API Designer

You can work offline (without a connection to the cloud) with API Designer to create APIs. Some functionality is missing due to the lack of a cloud connection.

About this task

API Designer provides the capability to work in offline mode without a connection to a management server. You can work on APIs with no management server, but some functions will be changed or missing.

The preferred method for working in API Designer is with a Cloud Connection to the management server when logging into API Designer. Working while connected to a management server provides full functionality, with automatic field completion. When working offline, you will need to remember the values to complete the fields that are backfilled from the management server.

Following are the differences when working in offline mode:

- There is no connection to a management server.
- There is no Provider Organization selection.
- Only the Develop APIs and Products tile may be selected. The Learn more and Connect menu tiles are not available.
- When creating an API, the Gateway Type defaults to V5 Gateway and the default policies are loaded.
- Stage and Publish for APIs and Products functions are not available.
- The Sandbox catalog is not available.
- When configuring a Security Definition, there are no selection lists available for the User Registry for a Basic Auth security definition or for the OAuth Provider for an OAuth 2 security definition. You will need to enter the User Registry and OAuth Provider values correctly as text.
- When adding custom Visibility and custom Subscribability values for a product, there is no selection list for the Organization field. You will need to enter the organization name correctly as text.

Procedure

To work in offline mode in API Designer, complete the following steps:

1. Log in to API Designer and open a directory location.
2. To start offline mode, choose Work Offline from the selector in the upper right corner of the Connect to Cloud screen.
3. The Home page appears with the Develop APIs and Products tile. Select Develop APIs and Products to start working in offline mode.
4. To return to online mode, choose Connect to Cloud in the upper right hand corner and select the management server to log in to.

Gathering support information in API Designer

When you report an issue with the API Designer application, enable debug mode and provide as much information as possible.

To report an issue with API Designer, enable debug mode, collect the recommended information, and submit it with your report. Including the information will help IBM resolve the problem more quickly.

- Enabling debug mode on Mac
- Enabling debug mode on Windows
- Enabling debug mode on Linux

- Information to gather for support

Enabling debug mode on Mac

Start API Designer with debug mode enabled:

1. Start a terminal or command prompt.
2. Navigate to the directory where API Designer is installed.
3. Run the following command to start API Designer in debug mode:

   ```
   DESIGNER_DEBUG=true open API/Designer.app/
   ```

   If you add **`DESIGNER_DEBUG=true`** to your .bash_profile, then every time you start API Designer it will open in debug mode.

Enabling debug mode on Windows

Add a Windows environment variable that enables debug mode for API Designer:

1. Type "Environment" in the Windows search bar (on the task bar) and select Edit environment variables for your account.
2. In the Environment Variables dialog box, click New in the "User variables" section.
3. In the New User Variable dialog box, add the following variable settings and then click OK:
   - Variable name: `DESIGNER_DEBUG`

   - Variable value: `True`

4. Click OK to save your change.

Enabling debug mode on Linux

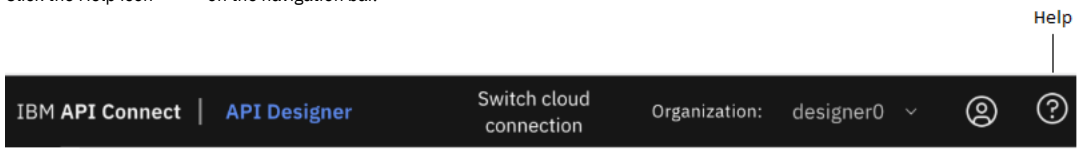
To start API Designer with debug mode enabled, add the following command to your .bash_profile:

```
EXPORT DESIGNER_DEBUG=true
```

Information to gather for support

Gather the following information about your problem and include it with your report.

- Operating system where you are running API Designer (Mac, Windows, or Linux)
- Product versions for API Designer and the API Manager that is connects to:
  1. Open API Designer.
  2. Click the Help icon on the navigation bar.

  3. On the Help page, click the Product information tile.
  4. On the Product information page, note down the following values for the following labels on the page:
     - Connected to API Manager Product version
     - Package
- HAR file
  The HAR file contains the information about your browser's interaction with the web, and can be helpful in determining why your problem is happening. Complete the following steps to export the HAR file from the browser where you use API Designer:

  1. Open API Designer.
  2. Open the browser's developer console; for example in Google Chrome, click > More tools > Developer tools.
  3. In the developer console, click the Network tab.
  4. Click to download the HAR file.
- Are you using API Designer in Online mode or Offline mode?
  If you're using Online mode, are you connecting to a local test environment, or to an API Manager server?
- When does the problem happen?
  Provide reproducible steps with optional screen captures or video so the support team can recreate the problem.
- Supporting data.
  Provide any related data that would be useful while examining the problem; for example, a copy of the API or Product that you were using.
- How often does the problem happen?
  Is the problem intermittent? Does it happen constantly? Does it only happen with certain artifacts (for example, only with a particular API)? If the problem is associated with an artifact, try to include it with your report.

Using the developer toolkit command-line tool

The IBM® API Connect developer toolkit provides a command-line tool, **apic**, that you can use to perform all API Connect tasks. You can also use the command-line tool to script tasks such as continuous integration and delivery.

The command line tool is described in detail in the following subtopics. For summary reference material for each of the available commands for the developer toolkit, see Command-line tool reference for the developer toolkit.

- **Overview of the command-line tool**
  The IBM API Connect developer toolkit provides commands for cloud administration and API development and management.
- **Logging in to a management server**
  You log in to a management server from the command line by using the **apic login** command. The parameters that you supply determine the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in.
- **Logging in to a management server with an OIDC registry**
  Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.
- **Cloud administration commands**
  A summary of the cloud administration commands in the IBM API Connect developer toolkit.
- **API development and management commands**
  A summary of the core commands in the IBM API Connect developer toolkit.
- **Creating APIs and applications**
  You can develop API proxies and API implementations by using the developer toolkit. In the documentation, *API* refers to the API proxy and *application* refers to the API implementation.
- **Publishing APIs and applications**
  To publish APIs and applications by using the developer toolkit of IBM API Connect, you set configuration variables to define where you want to publish, log in to the target cloud platform, and then use the appropriate publishing commands.
- **Managing API Products**
  Use the **apic products** and **apic apis** commands to manage Products and APIs that have been published to IBM API Connect Catalogs. Use the **--scope space** option to manage Products and APIs that have been published to Spaces within Catalogs.
- **Working with Drafts**
  Co-locate your APIs and applications in your local source code control systems to support typical development activities such as commits, branching, merges, continuous integration, and so on. The developer toolkit provides the bridge from the developer's environment to the IBM API Connect runtime services.
- **Reading input from the command line**
  If a developer toolkit command takes a file as an input parameter, you can direct the command to read the input directly from the command line rather than supplying a separate file; this can be useful when writing scripts to automate command line operations, for example.
- **Scripting with the toolkit commands**
  The IBM API Connect developer toolkit provides commands for cloud administration and API development and management.
- **Working with OpenAPI extensions**
  Use the developer toolkit CLI **extensions** commands to manage OpenAPI extensions in your Catalogs or Spaces.
- **Using the toolkit CLI to execute API tests**
  The IBM API Connect developer toolkit provides CLI commands for automated API behavior testing.

Overview of the command-line tool

The IBM® API Connect developer toolkit provides commands for cloud administration and API development and management.

Command syntax

In general, commands have the following syntax:

apic *command*:*sub-command* [*argument*] [*options*]

where

- *command* is the command, usually the thing on which you are acting (for example, product, app, API, Catalogs, and so on).
- *sub-command* is the action to perform.
- *argument* is the argument, where applicable (for example,**catalog**).
- *options* are any number of command-line options, which have the form **--*option* [*value*]**. Options also have a short form with a single dash instead of a double dash.

For example, **apic apps:publish --server mgmnthost.com**.
For some commands, either the command or sub-command portion is optional. For example:

- **apic products:publish** is equivalent to **apic publish**.
- **apic products:list** is equivalent to **apic products**.

The **create** command has a slightly different syntax:

apic create:*type* [*options*]

Use the **-h** or **--help** option to view command help.
Note: The language in which the CLI help text, and other command response text, is displayed is determined by the locale setting on your local machine.

Viewing command line tool help

Display general command-line help information by entering the following command: **apic --help** or **apic -h**. Display help information for a specific **apic** command by entering the following command: **apic *command_name* --help** or **apic *command_name* -h**.

Viewing version information

Display the version of the command-line tool by entering the command: **apic --version**.

Setting the mode

The API Connect toolkit CLI runs commands on either the provider organization or on the consumer organization. When you run the command, you use the different mode options to identify the instance on which you want to run the command. The following list provides a summary of the modes that are available:

apim
: This mode applies to the administrative instances of the CLI. When you run commands in this mode, it uses the overall API Manager as its scope. This mode is the default mode.
An example of running a command in the apim mode:

```
apic extensions:list-all --mode apim --scope catalog --server server1 --catalog catalog1 --org myOrg --configured-gateway-service service1
```

portaladmin
: This mode applies to the developer portal instances of the CLI. This mode supports commands that you can use to replicate all or part of your Developer Portal environment.
An example of running a command in the portaladmin mode:

```
apic --mode portaladmin custom-theme:create-export --catalog catalog1 --org myOrg --server server1 --format json
```

consumer
: This mode applies to the consumer instances of the CLI. This command uses the consumer organizations that you have permission to access as its scope. You must specify this mode or set it as the default value for the command to use it.
An example of running a command in the consumer mode:

```
apic catalogs:list --mode consumer --scope catalog --server server1 --catalog catalog1 --org myOrg
```

To avoid having to supply the parameter on every consumer CLI command, you can set the **mode** configuration variable, as described in the next section.

Using configuration variables

You can set the values of commonly-used properties in configuration variables. In general, it's easier and more consistent to set configuration variables instead of specifying them using command-line options.
Note:
You can set a configuration variable locally (the default) to affect only the current LoopBack project, or globally (with command-line option **-g**), to affect all projects. The local value supersedes the global value. You can set local configuration variables only for LoopBack projects. When you set configuration variables for OpenAPI projects, they are always global.

The values of local configuration variables are stored in the *project-root*/.apiconnect/config file, where *project-root* is the project root directory. The values of global configuration variables are stored in the *user-home-dir*/.apiconnect/config file, where *user-home-dir* is the user's home directory.

Use the following commands to work with configuration variables:

- **apic config:get *varname*** - Get a configuration variable. Use **apic config** to display the values of all local configuration variables or **apic config -g** to display the values of all global configuration variables.
- **apic config:set *varname*** - Set or update the specified configuration variable.
- **apic config:delete *varname*** - Delete the specified configuration variable.
- **apic config:clear** - Delete all configuration variables.

You set configuration property values by using the **apic config:set** command. By setting configuration properties (for example **catalog** and **app**), you do not need to supply values for these options when you enter a command.

Additionally, you can use **apic properties** commands to work with configuration properties:

- **apic properties:clear** - Clear the configuration properties.
- **apic properties:create** - Augment the configuration properties with additional name/value pairs.
- **apic properties:delete** - Delete the configuration property.
- **apic properties:get** - Get the configuration property.
- **apic properties:list** - List the configuration properties.
- **apic properties:update** - Update the configuration property.

Note:
If you have an environment variable of the same name as a CLI configuration property then, by default, its value will override the value of the corresponding CLI configuration property for any CLI command at that scope.

For example, if you have defined an environment variable called **SPACE** then, by default, that value will be assumed for the value of the **--space** parameter in the following command, regardless of any **space** configuration property setting:

```
apic products:publish my_product.yaml --scope space
```

To prevent environment variables overriding CLI configuration properties, define an environment variable called **APIC_LOAD_FROM_ENV**, set to the value **false**.

The following table describes the configuration variables:

Table 1. Configuration variables

Variable name	Description	Use instead of (or override with) these flags...

Variable name	Description	Use instead of (or override with) these flags...
catalog	Default Catalog name for all commands that manage aspects of a Catalog. The Catalog value can be specified as either: • The full Catalog URI that includes server name and organization name: ***mgmt-server*/api/catalogs/*org-name*/*catalog-name***. In this case, you do not need to specify **--catalog**, **--organization**, or **--server** when you make CLI calls, since their values are included in the full Catalog URI. • The Catalog name, for example: **sandbox**. In this case, you still need to specify **--organization** and **--server** when you make CLI calls. Note: The Catalog name **apic-dev** is reserved for local testing.	• If Catalog URI used in variable assignment: **--catalog, --organization, --server** • If Catalog name used in variable assignment: **--catalog**
cloud	Default management server host name for cloud administration commands. Form: ***mgmt-server*/api/**.	**--server**
consumer	Default URI of an API consumer. Form: ***mgmt-server*/api/consumer-orgs/*org-name*/*catalog-name*/*consumer-org-name***, where *mgmt-server* is the management server, *org-name* is the organization name, *catalog-name* is the Catalog name, and *consumer-org-name* is the consumer organization name.	**--server, --organization, --catalog, --consumer**
mode	The default value of the **--mode** parameter for CLI commands. Set the value to **apim** or **consumer** depending on whether you want to run commands on a provider organization or a consumer organization. If you do not set this variable, and do not supply a **--mode** parameter on a command, the value **apim** is assumed.	--mode
org	Default org name for all commands that manage organizations. Form: ***mgmt-server*/api/orgs/*org-name***, where *mgmt-server* is the management server, *org-name* is the organization name. The org value can be specified as either: • The full org URI that includes the server name: ***mgmt-server*/api/orgs/*org-name***. In this case, you do not need to specify **--server** or **--organization** when you make CLI calls, since their values are included in the full org URI. • The org name, for example: **providerorg1**. In this case, you still need to specify **--server** when you make CLI calls.	• If org URI used in variable assignment: **--organization, --server** • If org name used in variable assignment: **--organization**
space	Default Space URI for all commands that manage aspects of a Space. Form: ***mgmt-server*/api/spaces/*org-name*/*catalog-name*/*space-name***, where *mgmt-server* is the management server, *org-name* is the organization name, *catalog-name* is the Catalog name, and *space-name* is the Space name. You can append the port number to the server name if it is not the default value of 443.	**--server, --organization, --catalog, --space**

To set configuration properties, enter the following command:

```
apic config:set name=value
```

where *name* is the name of the configuration property and *value* the value to assign to it.
For example:

```
apic config:set catalog=sandbox
```

Scripting commands

It's often helpful to automate a series of **apic** commands In a shell script. Since the **apic** tool first requires you to interactively accept the license, you must first use the following command:

```
apic --accept-license
```

Once you do that, your scripts can run non-interactively.
To disable collection of usage analytics, enter this command:

```
apic --live-help
```

Related information

- Command-line tool reference for the developer toolkit

Logging in to a management server

You log in to a management server from the command line by using the **apic login** command. The parameters that you supply determine the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in.

Attention: If you want to log in with an OIDC user registry or if you are an IBM Cloud Pak for Integration (CP4I) user, see Logging in to a management server with an OIDC registry.

1. Clear the credentials from any previous login by running the following command:

   ```
   apic client-creds:clear
   ```

   Clearing the credentials ensures that you do not inadvertently log in with the wrong set of credentials (for example, from a different product release or environment).

2. Set the credentials that you want to use now by running the following command:

```
apic client-creds:set path_to_credentials/credential.json
```

Where ***path_to_credentials*** is the location of the credential.json file that you want to use. For example:

```
apic client-creds:set /Users/local_user/credential.json
```

The parameters for the **apic login** command are as follows:

mgmt_endpoint_url
The URL depends on the type of organization that you are logging in with:

- Cloud admin organization or Provider organization, use one of the following URLs:
  - Platform API endpoint URL (note that if you want to run commands that relate to API governance, you must use the platform API endpoint URL to log in).
  - Management API manager URL.
- Consumer organizations, use the consumer API endpoint URL.

These endpoint URLs are configured during the installation of API Connect. If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in Viewing platform and UI endpoints, ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

user_id
The user ID you want to log in with. Depending on the tasks that you want to perform, this user ID might be any of the following:

- A user ID that is a member of the cloud administration organization. This is an ID that you could also use to log in to the Cloud Manager user interface.
- A user ID that is a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
- A user ID that is a member of a consumer organization. This is an ID that you could also use to log in to the Developer Portal.

password
The password associated with the supplied user ID.

realm
The *realm* parameter specifies the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in.
The format of the *realm* depends on the type of user, as follows:

- Member of the cloud administration organization:

  admin/*identity_provider*

  To determine the identity provider, see How to determine the identity provider.

- Member of a provider organization:

  provider/*identity_provider*

  To determine the identity provider, see How to determine the identity provider.

- Member of a consumer organization:

  consumer:*provider_org*:*catalog*/*identity_provider*

  where *provider_org* is the name of your provider organization, and *catalog* is the name of the Catalog in that provider organization.
  To determine the identity provider, see How to determine the identity provider.

  Important:
  - If you have installed toolkit credentials, as detailed in Installing the toolkit, then at any one time, you can use only one set of toolkit credentials for login to a management server from the toolkit CLI. If you want to log in to a different management server you must install the toolkit credentials from that management server.
    To revert to the default pre-supplied toolkit credentials for all login operations from the toolkit CLI, use the following command:

    ```
    apic client-creds:clear
    ```

  - If you log in to the CLI as a member of a consumer organization, you must supply the **--mode=consumer** parameter to the **apic login** command, and to all consumer commands. To avoid having to type the parameter every time, you can set the **mode** configuration variable, by entering the following command:

    ```
    apic config:set mode=consumer
    ```

You can also use the command interactively; enter **apic login** and you will be prompted for the values. For example:

```
apic login
Enter your API Connect credentials
Server? platform-api.myserver.com
Realm? provider/default-idp-2
Username? myuser
Password?
Logged into myserver.com successfully
```

Note: If you see a message stating that credentials cannot be found, download new credentials as explained in Downloading the toolkit.

How to determine the identity provider

If you want to log in as a member of the cloud administration organization, or as a member of a provider organization, you can help determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope scope --server mgmt_endpoint_url --fields title,realm
```

where *scope* has the value **admin** or **provider** depending on whether you want to log in as a member of the cloud administration organization, or as a member of a provider organization. The output lists the names and titles of all identity providers, for example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time.
The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**, and the default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
By default, API Connect creates a local user registry for user login for every context. The identity providers associated with these default registries are as follows:

Registry	Identity provider name
Cloud Manager Local User Registry (for login as a member of the cloud administration organization)	default-idp-1
API Manager Local User Registry (for login as a member of a provider organization)	default-idp-2
Sandbox Catalog User Registry (for login as a member of a consumer organization)	sandbox-idp

If you want to log in as a member of a consumer organization, see Logging in as a consumer to the Developer Portal by using the CLI for full details.

Logging out

To log out of a management server, use the following command:

```
apic logout --server mgmt_endpoint_url
```

Logging in to a management server with an OIDC registry

Logging in with an OIDC registry involves several steps, in which you obtain a temporary token and pass it to the toolkit to ensure it can communicate securely with the management server.

Attention: Logging in with an OIDC registry is supported for the developer toolkit (used with API Manager) and the Developer Portal admin toolkit, but is not supported for members of consumer organizations. If you are using IBM Cloud Pak for Integration (CP4I), review this topic for login instructions. If you want to log in using a non-OIDC registry, see Logging in to a management server.
To log in, complete the following steps:

1. Clear the credentials from any previous login by running the following command:

   ```
   apic client-creds:clear
   ```

   Clearing the credentials ensures that you do not inadvertently log in with the wrong set of credentials (for example, from a different product release or environment).

2. Set the credentials that you want to use now by running the following command:

   ```
   apic client-creds:set path_to_credentials/credential.json
   ```

   where ***path_to_credentials*** is the location of the credential.json file that you want to use. For example:

   ```
   apic client-creds:set /Users/local_user/credential.json
   ```

3. Log in by running the following command:

   ```
   apic login --server mgmt_endpoint_url --sso
   ```

   where***mgmt_endpoint_url*** is the endpoint URL. When you log in with a Cloud admin or Provider organization, specify one of the following URLs:
   - platform API endpoint URL
   - management API manager URL

   These endpoint URLs are configured during the installation of API Connect. If you have access to the Cloud Manager user interface, you can view the configured endpoint URLs as described in Viewing platform and UI endpoints, ignoring any segments at the end of the displayed URLs. If you are not sure of the endpoint URL, ask your administrator.

   CP4I: If you are using API Connect as a component of IBM Cloud Pak for Integration (CP4I), your administrator can provide the correct URL.
4. When the toolkit prompts for the context, type **admin** API Connect (administrators) or **provider** (everyone else):

   ```
   Context? provider
   ```

5. The server responds with the following message:

   ```
   Please copy and paste the url https://mgmt_endpoint_url/auth/manager/sign-in/?from=TOOLKIT to a browser to start the
   authentication process.
   Do you want to open the url in default browser? [y/n]: y
   ```

   Take one of the following actions:

   - If you want to use your default browser to log in to API Manager, type **y** and press Enter.
     The API Manager sign-in page opens in a new browser tab.

   - If you don't use to use your default browser, type **n** and press Enter.
     - Copy the URL from the command window.
     - Open a browser, paste the URL, and press Enter.

     The API Manager sign-in page displays.

6. On the API Manager sign-in page, select the OIDC registry and then log in to API Connect.
   CP4I: Select the Common Services User Registry.

After you log in, API Connect displays the "You are authenticated!" message and provides a temporary token. Copy the token.

7. Return to the command window. Paste the token at the **`API Key?`** prompt and press Enter.

When the token is validated and you are successfully logged in to the toolkit, the following message displays:

```
Warning: Using default toolkit credentials.
Logged into mgmt_endpoint_url successfully
```

Note: If you see a message stating that credentials cannot be found, download new credentials as explained in Downloading the toolkit.

Logging out

To log out of a management server, use the following command:

```
apic logout --server mgmt_endpoint_url
```

Cloud administration commands

A summary of the cloud administration commands in the IBM® API Connect developer toolkit.

Authenticating

Use the **`apic login`** command to authenticate to an API Manager service, and the **`apic logout`** command to remove your local authentication credentials.
Note: When you authenticate successfully, your credentials are stored, in plain text, in the file Linux .netrc or Windows _netrc. You should therefore set the file permissions in such a way that your credentials are not accessible by others.
For full details on how to log in to your management server from the CLI, see Logging in to the management server.

Configuring the command-line tool to use TLS certificates

API Manager uses TLS profiles to secure data transmission. For information on how to create a TLS profile in API Manager, see TLS profiles.

To configure the toolkit command-line tool to use certificates to communicate with an API Manager that has TLS profiles enabled, follow these steps:

For more information about the NODE_EXTRA_CA_CERTS environment variable, see Node.js documentation.

Note: When using any CLI command that applies directly to the configuration of your IBM API Connect cloud, you must supply the parameter setting **`--org admin`** for a command that requires an organization name as a parameter.

Command summary

The following tables summarize **`apic`** commands for cloud administration.

Table 1. Summary commands for cloud administrators

Command	Description	Sub-commands
`apic analytics-services`	Analytics Service commands	• **`apic analytics-services`** - Analytics Service collection operations. • **`apic analytics-services:clear`** - Clear the Analytics Services. • **`apic analytics-services:create`** - Create a Analytics Service. • **`apic analytics-services:delete`** - Delete the Analytics Service by name or ID. • **`apic analytics-services:get`** - Get the Analytics Service by name or ID. • **`apic analytics-services:list`** - List the Analytics Services. • **`apic analytics-services:update`** - Update the Analytics Service by name or ID.
`apic associates`	Associates commands	• **`apic associates`** - Associate collection operations. • **`apic associates:get`** - Get the Associate by name or ID. • **`apic associates:list`** - List the Associates.
`apic availability-zones`	Availability Zone commands	• **`apic availability-zones`** - Availability Zone collection operations. • **`apic availability-zones:clear`** - Clear the Availability Zones. • **`apic availability-zones:create`** - Create a Availability Zone. • **`apic availability-zones:delete`** - Delete the Availability Zone by name or ID. • **`apic availability-zones:get`** - Get the Availability Zone by name or ID. • **`apic availability-zones:list`** - List the Availability Zones. • **`apic availability-zones:update`** - Update the Availability Zone by name or ID.
`apic cloud-settings`	Cloud settings commands	• **`apic cloud-settings:get`** - Get the Cloud Setting. • **`apic cloud-settings:mail-server-configured`** - indicates whether a mail server has been configured. • **`apic cloud-settings:topology`** - Returns summary details of the configuration of your cloud topology. • **`apic cloud-settings:update`** - Update the Cloud Setting.

Command	Description	Sub-commands
`apic configured-gateway-services`	Commands for Configured Gateway Services in a Catalog or Space.	• `apic configured-gateway-services` - Configured Gateway Services operations. • `apic configured-gateway-services:clear` - Delete all Configured Gateway Services from a Catalog or Space. • `apic configured-gateway-services:create` - Configure a Gateway Service in a Catalog or Space. • `apic configured-gateway-services:delete` - Delete a Configured Gateway Service from a Catalog or Space. • `apic configured-gateway-services:get` - Obtain the details of a Configured Gateway Service in a Catalog or Space. • `apic configured-gateway-services:list` - List the Configured Gateway Services in a Catalog or Space.
`apic credentials`	Credential commands	• `apic credentials` - Application Credential collection operations. • `apic credentials:clear` - Clear the Application Credentials. • `apic credentials:create` - Create a Application Credential. • `apic credentials:delete` - Delete the Application Credential by name or ID. • `apic credentials:get` - Get the Application Credential by name or ID. • `apic credentials:list` - List the Application Credentials. • `apic credentials:reset` - Reset the client id and client secret. • `apic credentials:reset-client-secret` - Reset the client secret. • `apic credentials:update` - Update the Application Credential by name or ID. • `apic credentials:verify-client-secret` - Verify the client secret.
`apic extensions`	Extension commands	• `apic extensions` - Extension collection operations. • `apic extensions:document` - Get the document for an extension by id, or by name and version. • `apic extensions:get` - Get an extension by id, or by name and version. • `apic extensions:list` - List all versions of an extension given the name. • `apic extensions:list-all` - List all versions of all extensions.
`apic gateway-services`	Gateway service commands	• `apic gateway-services` - Gateway Service collection operations. • `apic gateway-services:clear` - Clear the Gateway Services. • `apic gateway-services:create` - Create a Gateway Service. • `apic gateway-services:delete` - Delete the Gateway Service by name or ID. • `apic gateway-services:get` - Get the Gateway Service by name or ID. • `apic gateway-services:list` - List the Gateway Services. • `apic gateway-services:reset-oauth-secret` - Reset the OAuth cryptographic • `apic gateway-services:update` - Update the Gateway Service by name or ID.
`apic integrations`	Integrations commands	• `apic integrations` - Integration collection operations. • `apic integrations:clear` - Clear the Integrations. • `apic integrations:clear-all` - Clear all of the Integrations in all of the collections. • `apic integrations:create` - Create a Integration. • `apic integrations:delete` - Delete the Integration by name or ID. • `apic integrations:get` - Get the Integration by name or ID. • `apic integrations:list` - List the Integrations. • `apic integrations:list-all` - List all Integrations in all collections. • `apic integrations:update` - Update the Integration by name or ID.
`apic keystores`	Keystores commands	• `apic keystores` - Keystore collection operations. • `apic keystores:clear` - Clear the Keystores. • `apic keystores:create` - Create a Keystore. • `apic keystores:delete` - Delete the Keystore by name or ID. • `apic keystores:get` - Get the Keystore by name or ID. • `apic keystores:list` - List the Keystores. • `apic keystores:update` - Update the Keystore by name or ID.
`apic mail-servers`	Mail server commands	• `apic mail-servers` - Mail Server collection operations. • `apic mail-servers:clear` - Clear the Mail Servers. • `apic mail-servers:create` - Create a Mail Server. • `apic mail-servers:delete` - Delete the Mail Server by name or ID. • `apic mail-servers:get` - Get the Mail Server by name or ID. • `apic mail-servers:list` - List the Mail Servers. • `apic mail-servers:test-connection` - Test a mail server connection. • `apic mail-servers:update` - Update the Mail Server by name or ID.
`apic notification-templates`	Notification template commands	• `apic notification-templates` - Notification Template collection operations. • `apic notification-templates:get` - Get the Notification Template by name or ID. • `apic notification-templates:list` - List the Notification Templates. • `apic notification-templates:list-all` - List all Notification Templates in all collections. • `apic notification-templates:update` - Update the Notification Template by name or ID.

Command	Description	Sub-commands
apic permissions	Permissions commands	• **apic permissions** - Permission collection operations. • **apic permissions:get** - Get the Permission by name or ID. • **apic permissions:list** - List the Permissions. • **apic permissions:list-all** - List all Permissions in all collections.
apic portal-services	Portal services commands	• **apic portal-services** - Portal Service collection operations. • **apic portal-services:clear** - Clear the Portal Services. • **apic portal-services:create** - Create a Portal Service. • **apic portal-services:delete** - Delete the Portal Service by name or ID. • **apic portal-services:get** - Get the Portal Service by name or ID. • **apic portal-services:list** - List the Portal Services. • **apic portal-services:update** - Update the Portal Service by name or ID.
apic services	Services commands	• **apic services** - Service collection operations. • **apic services:clear** - Clear the Services. • **apic services:clear-all** - Clear all Services in all collections. • **apic services:create** - Create a Service. • **apic services:delete** - Delete a Service. • **apic services:get** - Get the Service by name and version. • **apic services:list** - List the Services. • **apic services:list-all** - List all Services in all collections. • **apic services:update** - Update the Service by name and version.
apic space-settings	Space settings commands	• **apic space-settings:get** - Get the Space Setting. • **apic space-settings:update** - Update the Space Setting.
apic spaces	Space commands	• **apic spaces** - Space collection operations. • **apic spaces:clear** - Clear the Spaces. • **apic spaces:create** - Create a Space. • **apic spaces:delete** - Delete the Space by name or ID. • **apic spaces:get** - Get the Space by name or ID. • **apic spaces:list** - List the Spaces. • **apic spaces:transfer-owner** - Transfer owner to a new member. • **apic spaces:update** - Update the Space by name or ID.
apic tls-client-profiles	TLS client profile commands	• **apic tls-client-profiles** - TLS Client Profile collection operations. • **apic tls-client-profiles:clear** - Clear the TLS Client Profiles. • **apic tls-client-profiles:clear-all** - Clear all of the TLS Client Profiles in the collection. • **apic tls-client-profiles:create** - Create a TLS Client Profile. • **apic tls-client-profiles:delete** - Delete the TLS Client Profile by name or ID. • **apic tls-client-profiles:get** - Get the TLS Client Profile by name or ID. • **apic tls-client-profiles:list** - List the TLS Client Profiles. • **apic tls-client-profiles:list-all** - Lists all of the TLS Client Profiles in a collection. • **apic tls-client-profiles:update** - Update the TLS Client Profile by name or ID.
apic tls-server-profiles	TLS server profile commands	• **apic tls-server-profiles** - TLS Server Profile collection operations. • **apic tls-server-profiles:clear** - Clear the TLS Server Profiles. • **apic tls-server-profiles:clear-all** - Clear all of the TLS Server Profiles in the collection. • **apic tls-server-profiles:create** - Create a TLS Server Profile. • **apic tls-server-profiles:delete** - Delete the TLS Server Profile by name or ID. • **apic tls-server-profiles:get** - Get the TLS Server Profile by name or ID. • **apic tls-server-profiles:list** - List the TLS Server Profiles. • **apic tls-server-profiles:list-all** - List all of the TLS Server Profiles in the collection. • **apic tls-server-profiles:update** - Update the TLS Server Profile by name or ID.
apic truststores	Truststore commands	• **apic truststores** - Truststore collection operations. • **apic truststores:clear** - Clear the Truststores. • **apic truststores:create** - Create a Truststore. • **apic truststores:delete** - Delete the Truststore by name or ID. • **apic truststores:get** - Get the Truststore by name or ID. • **apic truststores:list** - List the Truststores. • **apic truststores:update** - Update the Truststore by name or ID.

Command	Description	Sub-commands
`apic user-registries`	User registry commands	• `apic user-registries` - User Registry collection operations. • `apic user-registries:clear` - Clear the User Registries. • `apic user-registries:create` - Create a User Registry. • `apic user-registries:delete` - Delete the User Registry by name or ID. • `apic user-registries:execute` - Execute a User Registry operation. • `apic user-registries:get` - Get the User Registry by name or ID. • `apic user-registries:list` - List the User Registries. • `apic user-registries:search` - search for users in a user registry. • `apic user-registries:test-connection` - Test a User Registry connection. • `apic user-registries:update` - Update the User Registry by name or ID.
`apic user-registry-settings`	User registry settings commands	• `apic user-registry-settings:get` - Get the User Registry Setting. • `apic user-registry-settings:update` - Update the User Registry Setting.
`apic users`	Users commands	• `apic users` - User collection operations. • `apic users:clear` - Clear the Users. • `apic users:create` - Create a User. • `apic users:delete` - Delete the User by name or ID. • `apic users:get` - Get the User by name or ID. • `apic users:list` - List the Users. • `apic users:request-password-reset` - send a password reset link. • `apic users:search-provider` - Search for provider users from an organization. • `apic users:update` - Update the User by name or ID.
`apic webhooks`	Webhooks commands	• `apic webhooks` - Webhooks collection operations. • `apic webhooks:get` - Get the Webhooks by name or ID. • `apic webhooks:list` - List the Webhooks. • `apic webhooks:update` - Update the Webhooks by name or ID.

API development and management commands

A summary of the core commands in the IBM® API Connect developer toolkit.

Authenticating

Use the `apic login` command to authenticate to an API Manager service, and the `apic logout` command to remove your local authentication credentials.
Note: When you authenticate successfully, your credentials are stored, in plain text, in the file Linux .netrc or Windows _netrc. Therefore, you should set the file permissions in such a way that your credentials are not accessible by others.
For full details on how to log in to your management server from the CLI, see Logging in to the management server.

Configuring the command-line tool to use TLS certificates

API Manager uses TLS profiles to secure data transmission. For information on how to create a TLS profile in API Manager, see TLS profiles.

To configure the toolkit command-line tool to use certificates to communicate with an API Manager that has TLS profiles that are enabled, follow these steps:

For more information about the NODE_EXTRA_CA_CERTS environment variable, see Node.js documentation.

Creating and managing local files

You create and work with API and Product definition YAML files locally before you stage them to API Manager.

To create a local API definition file, use the `apic create:api` command. To create a local Product definition file, use the `apic create:product` command.

Use the `apic apis` and `apic products` commands to list API Manager artifacts of the specified type.

To validate the syntactical correctness of a local API or Product definition file, use the `apic validate` command.

To create a draft API in API Manager from a local API or Product definition file, use the `apic draft-apis:create` and `apic draft-products:create` commands, respectively.

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before an API is validated, created in draft, staged, or published. For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Working with Catalogs and Spaces

To create a Catalog, use the `apic catalog:create` command. To view information on a Catalog, use the `apic catalog:get` command; to list all Catalogs that are contained in organizations that the currently authenticated user is a member of, use the `apic catalogs` command.

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see Using syndication in IBM API Connect.

To enable Spaces for a Catalog, use the following command:

```
apic catalogs:set catalog_name --spaces enabled
```

Use the toolkit `apic spaces` commands to create and manage Spaces:

- `apic spaces` - List Spaces contained in a Catalog.
- `apic:spaces create` - Create a Space in a Catalog.
- `apic:spaces get` - Get information on a Space in a Catalog.
- `apic:spaces set` - Set information on a Space in a Catalog.
- `apic:spaces delete` - Delete a Space in a Catalog.

Command summary

The following tables summarize `apic` commands for API development and management.

Table 1. Summary of general-purpose commands

Command	Description	Subcommands
`apic config`	List and manage configuration variables. For more information, see Using configuration variables. With no subcommand, lists values of defined configuration variables.	• `apic config` - Manage configuration variables • `apic config:clear` - Delete all configuration variables • `apic config:delete` - Delete a configuration variable • `apic config:get` - Get a configuration variable • `apic config:list` - List the application and global configuration variables • `apic config:set` - Set or update configuration variables
`apic create`	Create a draft API or Product definition YAML file.	• `create:api` - Create a draft API OpenAPI definition YAML file • `create:product` - Create a Product definition YAML file
`apic extensions`	Manage OpenAPI extensions in a Catalog. With no subcommand, lists the extensions in the production Catalog.	• `apic extensions` - Extension collection operations. • `apic extensions:clear` - Delete all versions of an extension given the name. • `apic extensions:clear-all` - Delete all versions of all extensions. • `apic extensions:clone` - Download all versions of all extensions to your local drive. • `apic extensions:create` - Create an extension. • `apic extensions:delete` - Delete an extension by ID, or by name and version. • `apic extensions:document` - Get the document for an extension by ID, or by name and version. • `apic extensions:get` - Get an extension by ID, or by name and version. • `apic extensions:list` - List all versions of an extension given the name. • `apic extensions:list-all` - List all versions of all extensions. • `apic extensions:update` - Update an extension by ID, or by name and version.

Command	Description	Subcommands
apic lb4	Create LoopBack® project and project artifacts. With no subcommand, creates a new LoopBack project. Note: To use the **apic lb4** command, you need to add a double dash, **--**, after the **lb4** subcommand to tell **apic** that the options behind the **--** are for the subcommand belonging to **lb4**. The second **--** is to signify the end of the options. For example, **$ apic lb4 controller -- --help** Note: You can use **$ apic lb4 --help** to show a list of commands that are available, and you can use **$ apic lb4 *subcommand* -- --help** to show the usage of a subcommand.	• **app** - Create a new LoopBack project (default) • **controller** - Add a controller to a LoopBack 4 application • **datasource** - Add a datasource to a LoopBack 4 application • **discover** - Discover models from relational databases • **example** - Download one of LoopBack 4 example projects • **extension** - Create a LoopBack 4 extension • **import-lb3-models** - Import one or more LoopBack 3 models to a LoopBack 4 application • **interceptor** - Generate interceptors • **model** - Add a new model to a LoopBack 4 application • **observer** - Generate life cycle observers for application start and stop • **openapi** - Generate controllers and models from OpenAPI specs • **relation** - Add a relation between two models in a LoopBack 4 application • **repository** - Add a new repository for a selected model to a LoopBack 4 application • **rest-crud** - Generate rest configs for model endpoints • **service** - Add a new remote or local service to a LoopBack 4 application • **update** - Check or update project dependencies of LoopBack modules
apic login	Log in to API Manager.	None. Specify server and credentials with the required flags: • **-p, --password** *password* • **-r, --realm** *realm*. Determines the identity provider that is used to authenticated the supplied user ID, and the scope of the tasks that can be performed after successful log in. • **-s, --server** *mgmt_endpoint_url*. Use the value of the platform API endpoint URL for this entry. • **-u, --username** *user_name* For full details on how to log in to your management server from the CLI, see Logging in to the management server.
apic logout	Log out from API Manager.	None. Specify server with the required flag: • **-s, --server** *mgmt_service*. You can append the port number to the server name if it is not the default value of 443.
apic validate	Validate API or Product definition YAML file.	None

Table 2. Summary of commands to manage APIs, Products, and Catalogs

Command	Description	Subcommands
apic apis	List and manage APIs that are staged or published to Catalog or Space. Default subcommand is **list-all**.	• **apic apis** - API collection operations • **apic apis:get** - Get the OpenAPI definition YAML file for an API by name and version, or by ID • **apic apis:list** - List an API by name • **apic apis:list-all** - List all APIs in a Catalog or Space • **apic apis:update** - Update an API by name
apic apps	List and manage developer applications that are registered in a consumer organization. Default subcommand is **list**.	• **apic apps** - Application collection operations • **apic apps:clear** - Clear all applications • **apic apps:create** - Create an application • **apic apps:delete** - Delete an application by name or ID • **apic apps:get** - Get an application object by name • **apic apps:list** - List all the applications in a consumer organization • **apic apps:update** - Update an application by name or ID
apic catalogs	List and manage Catalogs in a provider organization. Default subcommand is **list**.	• **apic catalogs:clear** - Clear all Catalogs from provider organization • **apic catalogs:create** - Create a Catalog • **apic catalogs:delete** - Delete a Catalog by name or ID • **apic catalogs:get** - Get a Catalog object by name or ID • **apic catalogs:list** - List all the Catalogs in a provider organization • **apic catalogs:transfer-owner** - Transfer the ownership of a Catalog to another user • **apic catalogs:update** - Update a Catalog by name or ID
apic catalog-settings	Manage the configuration settings for a Catalog	• **apic catalog-settings:get** - Get the Catalog settings. • **apic catalog-settings:update** - Update the Catalog settings.

Command	Description	Subcommands
apic drafts	List and manage the draft API and Product definitions in a provider organization. Default subcommand is **list**.	• **apic drafts** - Draft collection operations • **apic drafts:clear** - Delete all draft APIs and Products in a provider organization • **apic drafts:list** - List the draft APIs and Products in a provider organization
apic draft-apis	List and manage the draft API definitions in a provider organization. Default subcommand is **list-all**.	• **apic draft-apis** - Draft API collection operations • **apic draft-apis:clear** - Delete all versions of a draft API by name • **apic draft-apis:clear-all** - Delete all draft APIs in a provider organization • **apic draft-apis:clone** - Clone all draft APIs in a provider organization • **apic draft-apis:create** - Create a draft API • **apic draft-apis:delete** - Delete a Draft API by name and version • **apic draft-apis:document** - Get the Draft API document by name and version • **apic draft-apis:get** - Get the OpenAPI definition YAML file for an API by name and version • **apic draft-apis:list** - List all versions of a draft API by name • **apic draft-apis:list-all** - List all draft APIs in a provider organization • **apic draft-apis:update** - Update a draft API by name and version, supplying the revised OpenAPI definition file • **apic draft-apis:update-wsdl** - Update a draft SOAP API by name and version, supplying the revised WSDL definition file in a .wsdl or .zip format Note: There must be a match of the service names in the existing WSDL with the revised WSDL. If a match is found, the WSDL service is converted into a YAML file that is used to update the API. If no match is found, an error message is displayed. • **apic draft-apis:validate** - Validates a draft API • **apic draft-apis:wsdl** - Get the Draft API WSDL document by name and version
apic draft-products	List and manage the draft Product definitions in a provider organization. Default subcommand is **list-all**.	• **apic draft-products** - Draft Product collection operations • **apic draft-products:clear** - Delete all versions of a draft Product by name • **apic draft-products:clear-all** - Delete all draft Products in a provider organization • **apic draft-products:create** - Create a draft Product • **apic draft-products:delete** - Delete a draft Product by name and version • **apic draft-products:get** - Get the Draft Product by name and version • **apic draft-products:list** - List the Draft Products • **apic draft-products:list-all** - List all draft Products in a provider organization • **apic draft-products:publish-all** - Publishes the Draft Product • **apic draft-products:update** - Update a draft Product by name and version, supplying the revised definition file • **apic draft-products:validate** - Validates a draft Product
apic members	List and manage the members of a provider organization, consumer organization, Catalog, or Space. Default subcommand is **list**.	• **apic members** - Member operations • **apic members:clear** - Delete all members • **apic members:create** - Create a member • **apic members:delete** - Delete a member by name or ID • **apic members:get** - Get a member object by name or ID • **apic members:list** - List all members • **apic members:update** - Update the member by name or ID
apic member-invitations	List and manage member invitations. A member invitation is created when a user is invited to be a member of a provider organization, consumer organization, Catalog, or Space. Default subcommand is **list**.	• **apic member-invitations** - Member invitation collection operations. • **apic member-invitations:clear** - Delete all member invitations. • **apic member-invitations:create** - Create a member invitation. • **apic member-invitations:delete** - Delete a member invitation by name or ID. • **apic member-invitations:get** - Get a member invitation object by name or ID. • **apic member-invitations:list** - List all member invitations. • **apic member-invitations:update** - Update a member invitation by name or ID.
apic orgs	List and manage provider organizations, and the admin organization. Default subcommand is **list**.	• **apic orgs** - Organization collection operations • **apic orgs:clear** - Delete all organizations • **apic orgs:create** - Create an organization • **apic orgs:delete** - Delete an organization by name or ID • **apic orgs:get** - Get an organization object by name or ID • **apic orgs:list** - List all organizations • **apic orgs:transfer-owner** - Transfer ownership of an organization • **apic orgs:update** - Update an organization by name or ID
apic org-settings	Manage settings for provider organizations, and the admin organization.	• **apic org-settings:get** - Get the settings object for an organization • **apic org-settings:update** - Update the settings for an organization
apic policies	List and manage policies in a Catalog. Default subcommand is **list**.	• **apic policies** - Policy collection operations • **apic policies:clear** - Clear the Policies • **apic policies:clear-all** - Clear all Policies in all collections • **apic policies:create** - Create a Policy • **apic policies:delete** - Delete a Policy • **apic policies:get** - Get the Policy by name and version • **apic policies:list** - List the Policies • **apic policies:list-all** - List all Policies in all collections • **apic policies:update** - Update the Policy by name and version

Command	Description	Subcommands
`apic products`	List and manage Products that are staged or published to Catalog or Space. Default subcommand is `list-all`.	• `apic products` - Product collection operations • `apic products:clear` - Delete all versions of a Product by name • `apic products:clear-all` - Delete all Products • `apic products:delete` - Delete a Product by name and version • `apic products:execute-migration-target` - Products execute migration target operations • `apic products:get` - Get a Product object by name and version • `apic products:list` - List all versions of a Product by name • `apic products:list-all` - List all Products • `apic products:publish` - Publish a Product to a Catalog or Space, supplying the Product definition YAML file • `apic products:replace` - Replace a Product with another Product • `apic products:set-migration-target` - Set the target Product for migrating subscriptions from a Product • `apic products:supersede` - Supersede a Product with another Product • `apic products:update` - Update a Product by name and version, supplying the revised Product definition YAML file
`apic identity-providers`	View information about identity providers.	• `apic identity-providers` - Identity provider operations • `apic identity-providers:list` - List the identity providers.
`apic spaces`	List and manage Spaces contained in a Catalog. Default subcommand is `list`.	• `apic spaces` - Space collection operations • `apic spaces:clear` - Delete all Spaces from a Catalog • `apic spaces:create` - Create a Space in a Catalog • `apic spaces:delete` - Delete a Space from a Catalog, by name or ID • `apic spaces:get` - Get a Space object by name or ID • `apic spaces:list` - List all the Spaces in a Catalog • `apic spaces:transfer-owner` - Transfer the ownership of a Space to another user • `apic spaces:update` - Update a Space by name or ID
`apic subscriptions`	List and manage subscriptions in a Product or a Catalog. Default subcommand is `list`.	• `apic subscriptions` - Subscription collection operations • `apic subscriptions:clear` - Clear the Subscriptions • `apic subscriptions:create` - Create a Subscription • `apic subscriptions:delete` - Delete the Subscription by name or ID • `apic subscriptions:get` - Get the Subscription by name or ID • `apic subscriptions:list` - List the Subscriptions • `apic subscriptions:update` - Update the Subscription by name or ID

Table 3. Summary of other commands for API developers and managers

Command	Description	Subcommands
`apic consumer-orgs`	Manage consumer organizations	• `apic consumer-orgs` - Consumer Organization collection operations. • `apic consumer-orgs:clear` - Clear the Consumer Organizations. • `apic consumer-orgs:create` - Create a Consumer Organization. • `apic consumer-orgs:delete` - Delete the Consumer Organization by name or ID. • `apic consumer-orgs:get` - Get the Consumer Organization by name or ID. • `apic consumer-orgs:list` - List the Consumer Organizations. • `apic consumer-orgs:transfer-owner` - Transfer owner to a new member. • `apic consumer-orgs:update` - Update the Consumer Organization by name or ID.
`apic consumer-org-settings`	Manage consumer organization settings	• `apic consumer-org-settings:delete` - Delete the Consumer Organization Setting. • `apic consumer-org-settings:get` - Get the Consumer Organization Setting. • `apic consumer-org-settings:update` - Update the Consumer Organization Setting.
`apic groups`	Manage groups	• `apic groups` - Group collection operations. • `apic groups:clear` - Clear the Groups. • `apic groups:create` - Create a Group. • `apic groups:delete` - Delete the Group by name or ID. • `apic groups:get` - Get the Group by name or ID. • `apic groups:list` - List the Groups. • `apic groups:update` - Update the Group by name or ID.
`apic invitations`	Manage invitations. An invitation is created when a user is invited to be the owner of a provider organization, consumer organization, Catalog, or Space.	• `apic invitations` - Invitation collection operations. • `apic invitations:clear` - Delete all invitations. • `apic invitations:create` - Create an invitation. • `apic invitations:delete` - Delete an invitation by name or ID. • `apic invitations:get` - Get the details of an invitation by name or ID. • `apic invitations:list` - List all invitations. • `apic invitations:update` - Update an invitation by name or ID.

Command	Description	Subcommands
`apic member-invitations`	Manage member invitations. A member invitation is created when a user is invited to be a member of a provider organization, consumer organization, Catalog, or Space.	• `apic member-invitations` - Member invitation collection operations. • `apic member-invitations:clear` - Delete all member invitations. • `apic member-invitations:create` - Create a member invitation. • `apic member-invitations:delete` - Delete a member invitation by name or ID. • `apic member-invitations:get` - Get the details of a member invitation by name or ID. • `apic member-invitations:list` - List all member invitations. • `apic member-invitations:update` - Update a member invitation by name or ID.
`apic members`		• `apic members` - Member collection operations. • `apic members:clear` - Clear the Members. • `apic members:create` - Create a Member. • `apic members:delete` - Delete the Member by name or ID. • `apic members:get` - Get the Member by name or ID. • `apic members:list` - List the Members. • `apic members:update` - Update the Member by name or ID.
`apic registrations`		• `apic registrations` - Registration collection operations. • `apic registrations:clear` - Clear the Registrations. • `apic registrations:create` - Create a Registration. • `apic registrations:delete` - Delete the Registration by name or ID. • `apic registrations:get` - Get the Registration by name or ID. • `apic registrations:list` - List the Registrations. • `apic registrations:update` - Update the Registration by name or ID.
`apic role-defaults`		• `apic role-defaults` - Role Default collection operations. • `apic role-defaults:clear` - Clear the Role Defaults. • `apic role-defaults:create` - Create a Role Default. • `apic role-defaults:delete` - Delete the Role Default by name or ID. • `apic role-defaults:get` - Get the Role Default by name or ID. • `apic role-defaults:list` - List the Role Defaults. • `apic role-defaults:list-all` - List all Role Defaults in all collections. • `apic role-defaults:update` - Update the Role Default by name or ID.
`apic roles`		• `apic roles` - Role collection operations. • `apic roles:clear` - Clear the Roles. • `apic roles:create` - Create a Role. • `apic roles:delete` - Delete the Role by name or ID. • `apic roles:get` - Get the Role by name or ID. • `apic roles:list` - List the Roles. • `apic roles:update` - Update the Role by name or ID.
`apic tasks`		• `apic tasks` - Task collection operations. • `apic tasks:clear` - Clear the Tasks. • `apic tasks:create` - Create a Task. • `apic tasks:delete` - Delete the Task by name or ID. • `apic tasks:get` - Get the Task by name or ID. • `apic tasks:list` - List the Tasks. • `apic tasks:update` - Update the Task by name or ID.
`apic user-registries`		• `apic user-registries` - User Registry collection operations. • `apic user-registries:clear` - Clear the User Registries. • `apic user-registries:create` - Create a User Registry. • `apic user-registries:delete` - Delete the User Registry by name or ID. • `apic user-registries:execute` - Execute a User Registry operation. • `apic user-registries:get` - Get the User Registry by name or ID. • `apic user-registries:list` - List the User Registries. • `apic user-registries:search` - search for users in a user registry. • `apic user-registries:test-connection` - Test a User Registry connection. • `apic user-registries:update` - Update the User Registry by name or ID.
`apic user-registry-settings`		• `apic user-registry-settings:delete` - Delete the User Registry Setting. • `apic user-registry-settings:get` - Get the User Registry Setting. • `apic user-registry-settings:update` - Update the User Registry Setting.

Command	Description	Subcommands
`apic users`		• `apic users` - User collection operations. • `apic users:clear` - Clear the Users. • `apic users:create` - Create a User. • `apic users:delete` - Delete the User by name or ID. • `apic users:get` - Get the User by name or ID. • `apic users:list` - List the Users. • `apic users:request-password-reset` - send a password reset link. • `apic users:search-provider` - Search for provider users from an organization. • `apic users:update` - Update the User by name or ID.
`apic test`	Test your APIs	• `apic test {yaml-files}` - Define the tests. • `apic test {yaml-files} --insecureSkipVerify` - Test the APIs with certificate checks disabled. • `apic test {yaml-files} --certPath={location of certs folder}` - Specifies the location of the CA certificate if it is not available in the system keystore. • `api test {yaml-files} --mTLS --certPath={location of certs folder}` - Test the APIs with `mTLS` enabled. • `api test {yaml-files} --config-file={location of config file}` - Define the tests by passing the external configuration.

Creating APIs and applications

You can develop API proxies and API implementations by using the developer toolkit. In the documentation, *API* refers to the API proxy and *application* refers to the API implementation.

LoopBack® is a high-performance Node.js interaction-tier framework for APIs and micro-services. The LoopBack framework can be downloaded and installed using the **CLI + LoopBack + Designer** option for downloading the toolkit. See Installing the toolkit. To create a new LoopBack project, use the command `apic lb4`.
Note: IBM® API Connect is packaged with, and supports, LoopBack version 4. For more information on how to migrate your LoopBack 3 applications to LoopBack 4, see LoopBack 4 Migration Guide.
You can use the developer toolkit or LoopBack to create language-independent APIs by using OpenAPI to proxy to an existing backend implementation or to augment applications that are developed in other languages or frameworks such as Express®, Java™, Swift, Go, and others.

Creating development artifact definitions

Use the `apic create` command to create development artifacts, by using the following commands:

Command	Description
`apic create:api`	Create an OpenAPI definition.
`apic create:api --wsdl` *`filename`*	Create a SOAP API definition from a WSDL definition file, or a .zip file that contains the WSDL definition files for a service. The name and version of the generated API are obtained from the WSDL file. If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see Using an options file when importing a WSDL service.
`apic create:product`	Create an API Product definition.

Note: You can create an API or Product from an OpenAPI template file by using the `--template` *`template-name`* option.
You can also create Product and API definitions non-interactively by providing the `--title` option. This option sets several values that you can also customize with additional options; for example:

```
apic create:api --title Routes
apic create:product --title "Climb On"
```

You can also create the API and Product definitions at the same time:

```
apic create:api --title Routes --product "Climb On"
apic create:api --wsdl globalweather.wsdl --product "Weather Forecasting"
```

Alternatively, you can create APIs and then reference them when you create a new Product; for example:

```
apic create:api --title Routes
apic create:api --title Ascents
apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
```

Validating development artifact definitions

After you edit development artifacts or before you publish artifacts, best practice is to validate them; for example:

```
apic validate routes.yaml                    # Validate an API
apic validate climb-on.yaml                  # Validate the Product and its APIs
apic validate climb-on.yaml --product-only   # Validate the Product only (do not validate the referenced APIs)
```

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file,the `$ref` field is replaced with the contents of the target file before the draft API is created with the **apic drafts:validate** command. For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Developing LoopBack applications

After you've created a LoopBack application with the **apic lb4 app** command, you can add additional functionality to the application with the commands listed in the following table. Run these commands from the project root directory of the application. APIs created with LoopBack can be imported into API Manager.

Command	Description	Subcommands
apic lb4	Create LoopBack project and project artifacts. With no subcommand, creates a new LoopBack project. Note: To use the **apic lb4** command, you need to add a double dash, **--**, after the **lb4** subcommand to tell **apic** that the options behind the **--** are for the subcommand belonging to **lb4**. The second **--** is to signify the end of the options. For example, **$ apic lb4 controller -- --help**	• **app** - Create a new LoopBack project (default) • **controller** - Add a controller to a LoopBack 4 application • **datasource** - Add a datasource to a LoopBack 4 application • **discover** - Discover models from relational databases • **example** - Download one of LoopBack 4 example projects • **extension** - Create a LoopBack 4 extension • **import-lb3-models** - Import one or more LoopBack 3 models to a LoopBack 4 application • **interceptor** - Generate interceptors • **model** - Add a new model to a LoopBack 4 application • **observer** - Generate life cycle observers for application start and stop • **openapi** - Generate controllers and models from OpenAPI specs • **relation** - Add a relation between two models in a LoopBack 4 application • **repository** - Add a new repository for a selected model to a LoopBack 4 application • **rest-crud** - Generate rest configs for model endpoints • **service** - Add a new remote or local service to a LoopBack 4 application • **update** - Check or update project dependencies of LoopBack modules

Publishing APIs and applications

To publish APIs and applications by using the developer toolkit of IBM® API Connect, you set configuration variables to define where you want to publish, log in to the target cloud platform, and then use the appropriate publishing commands.

See the following sections for more information:

- Setting configuration variables
- Logging in to API Connect
- Publishing APIs

Setting configuration variables

The **apic config** command provides global and project-based configuration variables that specify the target Catalog for publishing APIs and applications. The values of these variables are stored in **~/.apiconnect/config** (for global variables) and ***project-dir*/.apiconnect** (for project variables). For a full list of configuration variables, see Overview of the command-line tool.

Set the **catalog** configuration variable to the URI of an API Connect Catalog to define a default Catalog target for all commands managing Catalogs. The **catalog** URI has the form:

https://*mgmt_endpoint_url*/api/catalogs/*org_name*/*catalog_name*

where *mgmt_endpoint_url* is the platform API endpoint URL, *org_name* is the provider organization name, and *catalog_name* is the Catalog name. The *mgmt_endpoint_url* portion sets the default value of the **--server** option, the *org_name* portion sets the default value of the **--org** option, and the *catalog_name* portion sets the default value of the **--catalog** option; you can override any of these values by including the corresponding option in a command.

Set the **space** configuration variable to the URI of an API Connect Space, to define a default Space target for all commands that manage Spaces. The **space** URI has the form:

https://*mgmt_endpoint_url*/api/spaces/*org_name*/*catalog_name*/*space_name*

where *mgmt_endpoint_url* is the platform API endpoint URL, *org_name* is the provider organization name, *catalog_name* is the Catalog name, and *space_name* is the name of the Space. The *mgmt_endpoint_url* portion sets the default value of the **--server** option, the *org_name* portion sets the default value of the **--org** option, the *catalog_name* portion sets the default value of the **--catalog** option, and the *space_name* sets the default value of the **--space** option; you can override any of these values by including the corresponding option in a command.

Although setting these configuration variables is not required, doing so simplifies commands that interact with API Connect clouds by providing default values for frequently used command-line options.

Here is an example of publishing with and without the **catalog** configuration variable set.

Without the configuration variable set:

```
apic products publish climb-on.yaml --server mgmnthost.com --org climbon --catalog sandbox
```

With the configuration variable set:

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
catalog: https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic products publish climb-on.yaml
```

You can override default values provided by the **catalog** configuration variable by providing one of the standard options with a different value. For example, use the **--catalog** option with the **apic products publish** command to specify the **qa** Catalog:

```
apic products publish climb-on.yaml --catalog qa
```

Don't forget about global configuration variables. If you use the same Catalog as the default target for multiple projects, set the value globally:

```
apic config:set --global catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
```

Note:
If you have an environment variable of the same name as a CLI configuration property then, by default, its value will override the value of the corresponding CLI configuration property for any CLI command at that scope.

For example, if you have defined an environment variable called **SPACE** then, by default, that value will be assumed for the value of the **--space** parameter in the following command, regardless of any **space** configuration property setting:

```
apic products:publish my_product.yaml --scope space
```

To prevent environment variables overriding CLI configuration properties, define an environment variable called **APIC_LOAD_FROM_ENV**, set to the value **false**.

Logging in to API Connect

Use the **apic login** and **apic logout** commands to manage your authentication. For details, see Logging in to a management server.

Publishing APIs

Publishing APIs to API Catalogs in API Connect clouds enables you to socialize the APIs by using the developer portal and secure them by using the Gateway.

An *API Product* (or simply *Product*) is used to compose APIs for publishing. API Product managers can use it to bundle one or more APIs together, control the visibility of the Product in the developer portal (for example, only allow partners x, y, and z to view and subscribe to the Product), and define Plans to provide consumption options. The Products that reference the APIs and define the consumption Plans are also the primary unit of lifecycle management for APIs.

Use the **apic products publish** command (equivalent to **apic products:publish**) to publish API Products to an API Connect cloud. The following example demonstrates how to create APIs composed by a Product, and publishing the Product and its APIs to a Catalog:

```
apic create:api --title Routes
apic create:api --title Ascents
apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/default-idp-2
apic products publish climb-on.yaml
```

For full details on how to log in to your management server from the CLI, see Logging in to a management server.

Add the **--stage** option to **apic publish** to stage the Product into a Catalog instead of publishing it. Products in a Catalog can be in the following states: staged, published, deprecated, or retired. For example:

```
apic products publish --stage climb-on.yaml
```

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see Using syndication in IBM API Connect.

If default configuration values have been set for the Space, Catalog, organization and management server, the following publish command could be used:

```
apic products publish --scope space product.yaml
```

where *product* is the name of the product that you want to publish.
Note: If the OpenAPI file that defines your API uses a **$ref** field to reference a fragment of OpenAPI code that is defined in a separate file, the **$ref** field is replaced with the contents of the target file before the Product that contains the API is staged or published with the **apic publish** command. For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Managing API Products

Use the **apic products** and **apic apis** commands to manage Products and APIs that have been published to IBM® API Connect Catalogs. Use the **--scope space** option to manage Products and APIs that have been published to Spaces within Catalogs.

Product management command summary

The following table summarizes the commands available for managing API Products. Usage examples follow.

Example command	Description
`apic config:set catalog=https://platform-api.myserver.com/api/catalogs/myorg/sandbox`	Set the default Catalog.
`apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/my-identity-provider`	Login into the management server. For full details on how to log in to your management server from the CLI, see Logging in to the management server.
`apic create:api --title Routes --product "ClimbOn"`	Create the Product and API.
`apic products:publish climbon.yaml`	Publish the Product to the default catalog. Add --stage argument to stage the Product.
`apic products:list-all --scope catalog`	List the Products in the default catalog.
`apic products:get climbon:1.0.0 --output -`	Display the Product's properties. Omit the output argument to download to a file.
`apic apis:list-all --scope catalog`	List the APIs in the Catalog.
`apic apis:get routes:1.0.0 --output -`	Get the API properties. Omit the output argument to download to a file.
`apic products:replace climbon:1.0.0 mapfile.txt`	Replace the Product indicated on the command line with the staged or published Product designated in the mapfile. This retires the replaced Product.
`apic products:supersede climbon:1.0.0 mapfile.txt`	Supersede the Product indicated on the command line with the staged or published Product designated in the mapfile. This deprecates the superseded Product.
`apic products:delete climbon:1.0.0`	Delete the Product from the Catalog. The Product must be either retired or deprecated.

Using the `products` and `apis` commands through a full lifecycle

This example shows a complex lifecycle where a new version of a Product and API replaces the original version at run time.

Note: If the OpenAPI file that defines your API uses a **`$ref`** field to reference a fragment of OpenAPI code that is defined in a separate file, the **`$ref`** field is replaced with the contents of the target file before the Product that contains the API is staged or published with the **apic publish** command. For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Set the default Catalog and login to the mgmnthost.com API Connect cloud

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox
apic login --username some-user --password some-password --server platform-api.myserver.com --realm my-identity-provider
```

For full details on how to log in to your management server from the CLI, see Logging in to the management server.

Create and publish an initial version

```
apic create:api --title Routes --version 1.0.0 --filename routes100.yaml
apic create:product --title "Climb On" --version 1.0.0 --apis routes100.yaml --filename climbon100.yaml
apic products:publish climbon100.yaml
```

Create a new version to fix a bug in the API, stage it to the Catalog

```
apic create:api --title Routes --version 1.0.1 --filename routes101.yaml
apic create:product --title "Climb On" --version 1.0.1 --apis routes101.yaml --filename climbon101.yaml
apic products:publish --stage climbon101.yaml
```

Inspect the Catalog

```
apic products:list-all --scope catalog
```

This produces a list of all Products in the catalog. Copy the ID of the `climbon:1.0.1` Product. The ID resembles the following example.

```
https:/server/api/catalogs/3eca6046-3c27-4ad/ab8772bf-0f65-45/products/8f854623-bda1-4f9
```

Create a mapping file

For example, if you are replacing a Product, the mapping file specifies the Product that you want to replace, and the mapping of the Plans from the source Product to the target Product. For example, if you are replacing `climbon:1.0.0` with `climbon:1.0.1` then `product_url` property specifies the URL of `climbon:1.0.0`. This file takes the following form:

```
product_url: https:/server/api/catalogs/{id}/products/{id}
plans:
- source: {source_plan_name_1}
  target: {target_plan_name_1}
- source: {source_plan_name_2}
  target: {target_plan_name_2}
            .
            .
            .
```

Note:

- The source and target Plan names can be the same or different.
- Every Plan in the Product you are replacing must be mapped to a Plan in the replacement Product.

"Hot-replace" version 1.0.0 with 1.0.1

```
apic products:replace climbon:1.0.1 PRODUCT_PLAN_MAPPING_FILE
```

After it is replaced in this way, a Product is retired. It is no longer active.

Note: The Product specified on the command line is the replacement Product. For example, if you are replacing **climbon:1.0.0** with **climbon:1.0.1** then the Product specified on the command line is **climbon:1.0.1**.

Supersede version 1.0.0 with 1.0.1

Rather than hot-replacing an existing Product with a new one (usually an updated version), you can supersede the existing Product with a new one. When you replace a Product, all external application subscriptions to that Product are automatically moved to the new Product. When you supersede a Product, the subscriptions are not automatically moved to the new Product, some action must be taken to subscribe external applications to the superseding Product.
The command to supersede a Product uses the same mapping file as the command to replace a Product. The name of the superseding Product is given on the command line.

```
apic products:supersede climbon:1.0.1 PRODUCT_PLAN_MAPPING_FILE
```

Once superseded, a Product is marked deprecated, meaning that no further subscriptions to the Product are allowed, although it is still active and existing subscriptions still work.
Note: The Product specified on the command line is the superseding Product. For example, if you are replacing **climbon:1.0.0** with **climbon:1.0.1** then the Product specified on the command line is **climbon:1.0.1**. The mapping file specifies the URL of the Product you are superseding.

Set a migration target

It is possible to set a migration target for the existing Product. This helps migration.
Use a command like the following to set the migration target.

```
apic products:set-migration-target climbon:1.0.0 PRODUCT_PLAN_MAPPING_FILE
```

Note: The Product specified on the command line is the Product from which you want to migrate subscriptions. The mapping file specifies the target Product for the subscriptions.
Once a migration target is set, external application developers will be able to migrate their application subscriptions through the Developer Portal with ease. It is also possible to execute a subscription migration using the following command.

```
apic products:execute-migration-target climbon:1.0.0
```

Delete a Product

When the replaced or superseded Product is no longer needed, it can be deleted.

```
apic products:delete climbon:1.0.0
```

Additional Product and API operations

In addition to the lifecycle management capabilities, you can download Products and APIs in a catalog or space using the **clone** sub-command:

Command	Description
apic products:clone	Download all Products and their APIs from the catalog or space.
apic apis:clone	Download all APIs from the catalog or space.

It can also be useful to clear all Products and their APIs from a Catalog, particularly for a development Catalog (you must provide the name of the Catalog as the value of the **--confirm** parameter):

```
apic products:clear --confirm catalog_name
```

where *catalog_name* is the name of the Catalog.
You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a subcatalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see Using syndication in IBM API Connect.

To manage the Products and APIs that have been published to a Space, include the **--scope space** option with the **apic products** and **apic apis** commands. For example, to list the Products that are contained in a Space called *flights*, use the following command:

```
apic products --scope space --space flights --catalog production --org climbonorg --server platform-api.myserver.com
```

Changing the lifecycle state of a Product in a Catalog or Space

To directly change the lifecycle state of a Product that has previously been staged or published to a Catalog or Space, complete the following steps:

- Enter the following command (the terminating hyphen character means that the command takes input from the command line):

  ```
  apic products:update product_name:version --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [--space space] -
  ```

  where:
  - *product_name* is the name of the Product whose lifecycle state you want to change.
  - *version* is the version of that Product.
  - *mgmt_endpoint_url* is the platform API endpoint URL.
  - *organization* is the name of the provider organization in which the Product was previously staged or published.
  - *scope* has one of the following values:
    - **catalog** if the Catalog does not have Spaces enabled.
    - **space** if the Catalog has Spaces enabled. If you specify **space** for the **--scope** parameter you must also supply the **--space** parameter.
  - *catalog* is the name of the Catalog in which the Product was previously staged or published.
  - (optional) *space* is the name of the Space. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command.

  The command returns:

  ```
  Reading PRODUCT_FILE arg from stdin
  ```

- Enter the following data, followed by a new line:

  ```
  state: new_state
  ```

  where *new_state* is the state that you want to change the Product to, and must have one of the following values.
  - staged

    - published
    - deprecated
    - retired
    - archived
- Press `CTRL D` to terminate the input. If the command is successful, the lifecycle state change is confirmed.
  For example:

```
apic products:update finance:1.0.0 --server https://myserver.com --org development --scope catalog --catalog sandbox -
Reading PRODUCT_FILE arg from stdin
state: published
finance:1.0.0    [state: published]   https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/0f0af980-f505-4f36-b09c-d7b1c9c1a2f2
```

Note:
The command will not succeed if the lifecycle state change you are attempting is not permitted.

For example, the following lifecycle state changes are permitted:

- staged to published
- deprecated to retired

The following lifecycle state changes are not permitted:

- published to staged
- retired to published

For full details, see The Product lifecycle.

To list all Products in a Catalog or Space, together with their lifecycle states, use the following command:

```
apic products:list-all --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [--space space]
```

For example:

```
apic products:list-all --server https://myserver.com --org development --scope catalog --catalog sandbox
graphql-services:1.0.0    [state: staged]      https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/7652d568-b396-4bfa-bf71-2f18cea63737
finance:1.0.0             [state: published]   https://myserver.com/api/catalogs/dce12994-a6a1-487b-83b6-c73bd8498799/006827d5-
9e82-427a-abe6-be5b126210f7/products/0f0af980-f505-4f36-b09c-d7b1c9c1a2f2
```

To find out the lifecycle state of a specific Product, use the following command:

```
apic products:get product_name:version --server mgmt_endpoint_url --org organization --scope scope --catalog catalog [--space
space] --fields state --output -
```

For example:

```
apic products:get finance:1.0.0 --server https://myserver.com --org development --scope catalog --catalog sandbox --fields
state --output -
state: published
```

Example scripts

A set of example toolkit scripts that demonstrate how to create and manage organizations, users, apps, Products and APIs are available at https://github.com/ibm-apiconnect/example-toolkit-scripts.

Working with Drafts

Co-locate your APIs and applications in your local source code control systems to support typical development activities such as commits, branching, merges, continuous integration, and so on. The developer toolkit provides the bridge from the developer's environment to the IBM® API Connect runtime services.

API Connect provides an online development capability called *Drafts* where you can define API and Product definitions. The **apic drafts** commands enable synchronization of Product and API artifacts between local source code control systems and drafts.

You can use **drafts** to clear and list your drafts. For example:

```
apic config:set catalog=https://platform-api.myserver.com/api/catalogs/climbon/sandbox    # set the default Catalog
apic login --username some-user --password some-password --server platform-api.myserver.com --realm provider/myldap  # login
into the management server

apic create:api --title routes --product ClimbOn
apic drafts                                          # list what is in drafts
apic drafts:clear --confirm drafts                   # clear drafts collection
apic drafts:list                                     # list the available drafts
```

For full details on how to log in to your management server from the CLI, see Logging in to the management server.

Reading input from the command line

If a developer toolkit command takes a file as an input parameter, you can direct the command to read the input directly from the command line rather than supplying a separate file; this can be useful when writing scripts to automate command line operations, for example.

Reading input from a file

Many developer toolkit commands that create or update new objects in API Connect take a file as an input parameter. For example:

- Register a new gateway service:

  ```
  apic gateway-services:create [flags] GATEWAY_SERVICE_FILE
  ```

- Update a provider organization:

  ```
  apic orgs:update [flags] ORG_FILE
  ```

- Create a new Catalog:

  ```
  apic catalogs:create [flags] CATALOG_FILE
  ```

- Create a new consumer organization:

  ```
  apic consumer-orgs:create [flags] CONSUMER_ORG_FILE
  ```

- Register a new developer application:

  ```
  apic apps:create [flags] APP_FILE
  ```

- Create a new email server:

  ```
  apic mail-servers:create [flags] MAIL_SERVER_FILE
  ```

- Update the list of applications that subscribe to a Product:

  ```
  apic subscriptions:update [flags] SUBSCRIPTION_FILE
  ```

You can format the input file as a YAML file (the default) or as a JSON file. To use a JSON file as input, include the `--format json` parameter.

Reading input from the command line

As an alternative to supplying an explicit file name to the command, you can supply a hyphen (-) in place of the file name parameter; the command will then read the file content directly from the command line as shown in the following examples.

- Register a new DataPower® Gateway (v5 compatible) gateway service:

  ```
  apic gateway-services:create --server platform-api.myserver.com --org admin --availability-zone availability-zone-default
  -
  Reading GATEWAY_SERVICE_FILE arg from stdin
  name: dpgw-service
  title: DataPower gateway service, compatible with v5
  gateway_service_type: datapower-gateway
  endpoint: 'https://mygwhost.com:3000'
  api_endpoint_base: 'https://mygwhost.com:9443'
  sni:
    - host: '*'
      tls_server_profile_url: https://platform-api.myserver.com/api/orgs/75203636-f038-4287-a732-24af4bf7059d/tls-server-
  profiles/3c0a0e93-6aa4-4288-b09c-eccf4901b104
  visibility:
    type: public
  ```

  Note: You can obtain the `tls_server_profile_url` property value for a TLS server profile by using the following command, which lists the URLs for all TLS server profiles:

  ```
  apic tls-server-profiles --org organization_name --server mgmt_endpoint_url
  ```

- Create a new provider organization:

  ```
  apic orgs:create --server platform-api.myserver.com -
  Reading ORG_FILE arg from stdin
  name: development
  title: Development organization
  owner_url: https://platform-api.myserver.com/api/user-registries/1bbbd414-22f1-47cf-8eac-2050530d29a7/c082d755-866e-4394-
  959f-fbb264c6c3a1/users/2db491c7-2d0f-4f60-a8ae-f38d07481f21
  ```

  Note: You can obtain the `owner_url` property value by using the following command, which returns the `url` property for a specific user:

  ```
  apic users:get username --user-registry user_registry_name --server mgmt_endpoint_url --org organization_name --fields url
  --output -
  ```

- Create a new Catalog:

  ```
  apic catalogs:create --server platform-api.myserver.com --org myorg -
  Reading CATALOG_FILE arg from stdin
  name: production
  title: Production Catalog
  summary: Catalog containing APIs in production use
  ```

- Create a new consumer organization:

  ```
  apic consumer-orgs:create --server platform-api.myserver.com --catalog sandbox --org myorg -
  Reading CONSUMER_ORG_FILE arg from stdin
  name: finance-apps
  title: Developers of finance applications
  owner_url: https://platform-api.myserver.com/api/user-registries/a1fb8159-fda1-410f-93ab-a0ea5fd04535/70ce7ec3-e83f-4c87-
  9f3c-4662ad108bbc/users/fddd5df5-c178-4a34-ab92-0a48344d5c9b
  ```

Note: You can obtain the `owner_url` property value by using the following command, which returns the `url` property for a specific user:

```
apic users:get username --user-registry user_registry_name --server mgmt_endpoint_url --org organization_name --fields url
--output -
```

- Register a new developer application:

```
apic apps:create --server consumer-api.myserver.com --org my-consumer-org --mode consumer -
Reading APP_FILE arg from stdin
name: finance
title: Finance application
summary: Mobile app for personal finance management
```

- Create a new email server:

```
apic mail-servers:create --org admin --server platform-api.myserver.com  -
Reading MAIL_SERVER_FILE arg from stdin
title: My email server
name: my-email-server
host: smtp.myemail.com
port: 20
credentials:
  username: me@myemail.com
  password: password
```

- Create a new subscription to subscribe an application to a Product:

```
apic subscriptions:create --app myapp --catalog sandbox --org myorg --server platform-api.myserver.com --consumer-org my-
consumer-org -
Reading SUBSCRIPTION_FILE arg from stdin
product_url: https://platform-api.myserver.com/api/catalogs/960733c1-d7f7-4b90-9bc7-69dbfbe4ce31/ca34537d-adf4-4320-8495-
a7feaa62d679/products/638015c6-a8b0-452e-841f-2ac441ab6962
plan: default-plan
```

Note: You can obtain the `product_url` property value for a Product by using the following command, which lists the URLs for all versions of a Product given the Product name:

```
apic products:list product_name --scope catalog --catalog catalog_name --org provider_org_name --server mgmt_endpoint_url
```

Related concepts

- Overview of the command-line tool

Related reference

- Logging in to a management server

Related information

- Command-line tool reference

Scripting with the toolkit commands

The IBM® API Connect developer toolkit provides commands for cloud administration and API development and management.

Scripting commands

It's often helpful to automate a series of `apic` commands In a shell script. Since the `apic` tool first requires you to interactively accept the license, you must first use the following command:

```
apic --accept-license
```

Once you do that, your scripts can run non-interactively.
Toolkit command scripts allow you to automate many of the tasks required to set up and maintain your API products. A set of example scripts are available at **https://github.com/ibm-apiconnect/example-toolkit-scripts**.

These scripts demonstrate how to perform the following tasks.

- Create a Provider Organization
- Configure a catalog to use one or more gateways
- Create draft APIs and Products
- Publish Products
- Replace an existing published product with a new version
- Delete a product
- Create a Consumer Organization
- Create a new consumer app
- Subscribe the new app to an existing published product
- Stage a new product version
- Supercede the existing published product with the new one
- Migrate existing subscriptions from the old product to the new product

Related information

- Command-line tool reference for the developer toolkit

Working with OpenAPI extensions

Use the developer toolkit CLI **extensions** commands to manage OpenAPI extensions in your Catalogs or Spaces.

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API, depending on the version of IBM® API Connect you are using. An extension is imported into a Catalog, or as Space ion a Catalog, then added to the API schema.

The commands described here are for importing OpenAPI extensions into a Catalog or Space, and viewing and managing them. After you have imported an OpenAPI extension, you can reference it in your API definitions; for details, see Referring to an extension in an API definition.

Note: If Spaces are enabled in a Catalog, an OpenAPI extension that you import into one Space is imported into **all** Spaces; you cannot import an OpenAPI extension into an individual Space in the Catalog. Any subsequent updates are also applied to all Spaces.

Command name	Action	Syntax
extensions clear	The **extensions:clear** command deletes all versions of an extension given the extension name. The parameters are as follows: --scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available: **`catalog`** The command is to be applied to Catalog **`space`** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **`space`** and supply the **`--space`** parameter. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **`--space`** parameter is required if the Catalog has Spaces enabled, in which case you must also include **`--scope`** **`space`** in the command. --configured-gateway-service <*service_name*> Specifies the name of the gateway service. --confirm <*catalog_name*> Confirms the clear of the extension. <*extension_name*> Specifies the name of the existing extension. Note: This is the extension name, not the file name. --org or -o <*organization_name*> Specifies a single organization with the organization name. --server or -s <*management_server_endpoint*> Specifies the server endpoint.	**`apic extensions:clear --scope catalog|space --catalog_|_-c catalog_name extension_name --server_|_-s management_server_endpoint --configured-gateway-service gw_service_name --confirm catalog_name --org_|_-o org_name [--space space]`** Example: **`apic extensions:clear --scope catalog --catalog catalog1 extension1 --server endpoint1 --configured-gateway-service gw_service1 --confirm catalog1 --org my_org`** This example deletes all versions of the **`extension1`** extension that are in **`catalog1`** of the **`my_org`** organization, and are paired with **`endpoint1`**.

Command name	Action	Syntax
extensions clear-all	The **extensions:clear-all** command deletes all of the extensions in the specified Catalog or Space. The parameters are as follows: --scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. --configured-gateway-service <*service_name*> Specifies the name of the gateway service. --confirm <*catalog_name*> Confirms the clear of the extension. --org or -o <*organization_name*> Specifies a single organization with the organization name. --server or -s <*management_server_endpoint*> Specifies the server endpoint.	**apic extensions:clear-all** **--scope** **catalog|space** **--catalog | -c** ***catalog_name*** **--server | -s** ***mgt_server_endpoint*** **--configured-gateway-service** ***gw_service_name*** **--confirm** ***catalog_name*** **--org | -o** ***org_name*** **[--space** ***space*****]** Example: **apic extensions:clear-all --scope catalog** **--catalog catalog1 --server endpoint1** **--configured-gateway-service gw_service1** **--confirm catalog1 --org my_org** This example deletes all the extensions that are in **catalog1** of the **my_org** organization, and are paired with **endpoint1**.
extensions clone	The **extensions:clone** command creates a local definition file for each extension. The parameters are as follows: --scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **org** Sets the scope to the organization. **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. --org or -o <*organization_name*> Specifies a single organization with the organization name. --server or -s <*management_server_endpoint*> Specifies the server endpoint. --configured-gateway-service <*service_name*> Specifies the name of the gateway service.	**apic extensions:clone** **--scope** **catalog|space** **--catalog | -c** ***catalog_name*** **--org | -o** ***org_name*** **[--space** ***space*****]** **--server | -s** ***mgt_server_endpoint*** **--configured-gateway-service** ***gw_service_name*** Example: **apic extensions:clone --scope catalog** **--catalog catalog1 --org my_org --server endpoint1** **--configured-gateway-service gw_service_1** This example clones the extensions that are in **catalog1** of the **my_org** organization, and are paired with **endpoint1**.

Command name	Action	Syntax
extensions create	The **extensions:create** command creates an extension in a Catalog or Space, given an extension definition file. The parameters are as follows: --scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. <*extension_file*> Specifies the name of the extension file that contains the new information for your extension. Note: This is the OpenAPI file name, not the name of the extension. --server or -s <*management_server_endpoint*> Specifies the server endpoint. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. --org or -o <*organization_name*> Specifies a single organization with the organization name. --configured-gateway-service <*service_name*> Specifies the name of the gateway service.	**apic extensions:create** --scope **catalog|space** *extension_file* --server | -s *mgt_server_endpoint* --catalog | -c *catalog_name* **[--space** *space***]** --org | -o *org_name* --configured-gateway-service *gw_service_name* Example: **apic extensions:create --scope catalog filename.yaml --server endpoint1 --catalog catalog1 --org my_org --configured-gateway-service gw_service_1** This example creates an extension in **catalog1** of the **my_org** organization, paired with **endpoint1**, using the **filename.yaml** OpenAPI extension definition file.
extensions delete	The **extensions:delete** command deletes a specific version of an extension. The parameters are as follows: --scope Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. --server or -s <*management_server_endpoint*> Specifies the server endpoint. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. <*extension_name*>:<*version_number*> Specifies the name and version number of the extension. Note: *extension_name* is the extension name, not the file name. --org or -o <*organization_name*> Specifies a single organization with the organization name. --configured-gateway-service <*service_name*> Specifies the name of the gateway service.	**apic extensions:delete** --scope **catalog|space** *extension_name*:*version_number* --catalog | -c *catalog_name* --org | -o *org_name* **[--space** *space***]** --server | -s *mgt_server_endpoint* **--configured-gateway-service** *gw_service_name* Example: **apic extensions:delete --scope catalog myextension:1.0.0 --catalog catalog1 --org orgmain --server endpoint1 --configured-gateway-service gw_service_1** This example deletes **myextension** version **1.0.0** that is in **catalog1** of the **orgmain** organization, and is paired with **endpoint1**.

Command name	Action	Syntax
extensions get	The **extensions:get** command retrieves the definition file for a specific version of an extension. The parameters are as follows: --scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **org** Sets the scope to the organization. **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. --server or -s <*management_server_endpoint*> Specifies the server endpoint. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. <*extension_name*>:<*version_number*> Specifies the name and version number of the extension. Note: *extension_name* is the extension name, not the file name. --org or -o <*organization_name*> Specifies a single organization with the organization name. --configured-gateway-service <*service_name*> Specifies the name of the gateway service.	**apic extensions:get** [--scope](#) **catalog|space** ***extension_name*:*version_number*** [--catalog | -c](#) ***catalog_name*** [--org | -o](#) ***org_name*** **[--space *space*]** [--server | -s](#) ***mgt_server_endpoint*** **--configured-gateway-service *gw_service_name*** Example: **apic extensions:get --scope catalog myextension:1.0.0 --catalog catalog1 --org orgmain --server endpoint1 --configured-gateway-service gw_service_1** This example gets **myextension** version **1.0.0** that is in **catalog1** of the **orgmain** organization, and is paired with **endpoint1**.
extensions list	The **extensions:list** command lists all versions of an extension given the extension name. The parameters are as follows: --scope Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available: **catalog** The command is to be applied to Catalog **org** Sets the scope to the organization. **space** The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to **space** and supply the **--space** parameter. --server or -s <*management_server_endpoint*> Specifies the server endpoint. --catalog or -c <*catalog_name*> Specifies a single Catalog with the Catalog name. --space Specifies the name of a Space in a Catalog. The **--space** parameter is required if the Catalog has Spaces enabled, in which case you must also include **--scope space** in the command. <*extension_name*> Specifies the name of the existing extension. Note: This is the extension name, not the file name. --org or -o <*organization_name*> Specifies a single organization with the organization name. --configured-gateway-service <*service_name*> Specifies the name of the gateway service.	**apic extensions:list** [--scope](#) **catalog|space** ***extension_name*** [--catalog | -c](#) ***catalog_name*** [--org | -o](#) ***org_name*** **[--space *space*]** [--server | -s](#) ***mgt_server_endpoint*** **--configured-gateway-service *gw_service_name*** Example: **apic extensions:list --scope catalog my_extension --catalog catalog1 --org my_org --server endpoint1 --configured-gateway-service gw_service_1** This example lists all versions of the extension **my_extension** that are paired with **endpoint1**, are in **catalog1**, and in the **my_org** organization.

Command name	Action	Syntax

extensions list-all	The **extensions:list-all** command lists all of the extensions that are available in a Catalog or Space. This is the default command if you enter only `apic extensions`.
	
	The parameters are as follows:
	
	--scope
	Specifies whether you want to clone the extensions in a Catalog or Space. The following options are available:
	
	`catalog`
	The command is to be applied to Catalog
	`org`
	Sets the scope to the organization.
	`space`
	The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to `space` and supply the `--space` parameter.
	
	--server or -s <*management_server_endpoint*>
	Specifies the server endpoint.
	--catalog or -c <*catalog_name*>
	Specifies a single Catalog with the Catalog name.
	--space
	Specifies the name of a Space in a Catalog. The `--space` parameter is required if the Catalog has Spaces enabled, in which case you must also include `--scope`
	`space` in the command.
	--org or -o <*organization_name*>
	Specifies a single organization with the organization name.
	--configured-gateway-service <*service_name*>
	Specifies the name of the gateway service.
		`apic extensions:list-all --scope catalog|space`
		`--catalog | -c catalog_name`
		`--org | -o org_name`
		`[--space space]`
		`--server | -s mgt_server_endpoint`
		`--configured-gateway-service gw_service_name`
		
		Example:
		
		`apic extensions:list-all --scope catalog`
		`--catalog catalog1 --org my_org --server endpoint1`
		`--configured-gateway-service gw_service1`
		
		This example lists the extensions that are paired with `endpoint1`, are in `catalog1`, and in the `my_org` organization.

extensions update	The **extensions:update** command replaces the information of an existing version of an extension with the information in an external file.
	
	The parameters are as follows:
	
	--scope
	Specifies whether you want to clear the extensions in a Catalog or Space. The following options are available:
	
	`catalog`
	The command is to be applied to Catalog
	`space`
	The command is to be applied to a Space in a Catalog. When Spaces are enabled in a Catalog, you must set the scope to `space` and supply the `--space` parameter.
	
	--server or -s <*management_server_endpoint*>
	Specifies the server endpoint.
	--catalog or -c <*catalog_name*>
	Specifies a single Catalog with the Catalog name.
	--space
	Specifies the name of a Space in a Catalog. The `--space` parameter is required if the Catalog has Spaces enabled, in which case you must also include `--scope`
	`space` in the command.
	--org or -o <*organization_name*>
	Specifies a single organization with the organization name.
	--configured-gateway-service <*service_name*>
	Specifies the name of the gateway service.
	<*extension_name*>:<*version_number*>
	Specifies the name and version number of the extension.
	Note: *extension_name* is the extension name, not the file name.
	<*extension_file*>
	Specifies the name of the extension file that contains the new information for your extension.
	Note: This is the OpenAPI file name, not the name of the extension.
		`apic extensions:update --scope catalog|space`
		`extension_name:version_number`
		`extension_file`
		`--catalog | -c catalog_name`
		`--org | -o org_name`
		`[--space space]`
		`--server | -s mgt_server_endpoint`
		`--configured-gateway-service gw_service_name`
		
		Example:
		
		`apic extensions:update`
		`--scope catalog extension1:1.0.0 extension2.yaml`
		`--catalog catalog1 --org my_org --server endpoint1`
		`--configured-gateway-service gw_service1`
		
		This example updates `extension1` version `1.0.0` that is paired with `endpoint1`, in `catalog1`, and in the `my_org` organization, from the file `extension2.yaml`.

Using the toolkit CLI to execute API tests

The IBM® API Connect developer toolkit provides CLI commands for automated API behavior testing.

About this task

Use CLI commands to execute API tests.
Note: In order to execute an API for testing purpose, you must make it online. To make the API online and auto-publish your API document, set the **Auto-publish** toggle on the Preferences page to **On** position. For more information, see Activating an API.

Procedure

1. Install the toolkit: Setting up the API Connect toolkit.
2. Open Terminal.
3. Create a YAML file to define your tests. For more information, see API development and management commands.
4. Run the CLI with the following syntax to execute the tests:

   ```
   apic test yaml-files
   ```

   Where, ***yaml-files*** specifies the location of the YAML file. For example, `apic test test1.yaml`
   Note: Cert files must be named in the following format:
   - `tls.crt` - public key
   - `tls.key`- private key
   - `ca.crt`- CA certificate

   For more information about how the test is defined as a YAML file, see Test definition syntax.
   For more information about the assigning variables on a YAML configuration file, see Variable assignment in YAML configuration.

- **Test definition syntax**
  Information about how the test is defined as a YAML file.
- **Variable assignment in YAML configuration**
  You can define variables within a YAML configuration file using the CLI tool. These variables are used to configure various aspects of the tool.

Test definition syntax

Information about how the test is defined as a YAML file.

The test is defined as a YAML file with the following sections:

- `config` - Displays the default values for the variables that are specified in the file.
- `steps` - Displays the flow for the test case and runs through the list of steps in order. Each step has a common set of parameters.

The `type` attribute in the test definition YAML file describes the type of `steps`.

```
steps:
  - type: request
    method: get
    url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/bank/branches
    var: branches
    mode: json
  - type: assert-is
    expression: branches
    value: array
  - type: assert-exists
    expression: branches.[1]
    stoponfail: true
  - type: assert-is
    expression: branches.[0].type
    value: string
    stoponfail: true
  - type: assert-is
    expression: branches.[0].address.city
    value: string
    stoponfail: true
  - type: assert-compares
    expression1: branches.[0].type
    expression2: branches.[7].type
  - type: request
    method: get
    url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/orders/order/AB1234
    var: payload
    mode: json
  - type: assert-equals
    expression: payload_response_statusCode
    value: 200
    ifexists: false
    stoponfail: false
  - type: assert-equals
    expression: payload_response_header_Content-Type
    value: application/json
  - type: assert-in
    expression: payload_response_header_Content-Type
    values:
      - application/json
      - application/swagger+yaml
  - type: assert-contains
    expression: payload_response_header_Content-Type
    value: app
  - type: assert-matches
    expression: payload_response_header_Content-Type
    value: ^[a-z]+/json
  - type: assert-exists
    expression: "payload"
```

```
    stoponfail: false
  - type: assert-is
    expression: "payload.tracking_reference"
    value: "string"
    ifexists: "false"
    stoponfail: false
  - type: assert-exists
    expression: payload.shipped_at
    stoponfail: false
  - type: assert-exists
    expression: payload.status
    stoponfail: false
  - type: assert-exists
    expression: payload.created_at
    stoponfail: false
```

The following table summarizes the set of parameters supported by the **type** attribute.

Table 1. Summary of the parameters supported by the **type** attribute

Parameters	Description	Sub-attributes

request	Make an HTTP request.	• **method** - defines the HTTP method to use.
		• **url** - defines the URL to make the request to.
		• **params** - defines the key value mapping of parameters to send.
		• **headers** - defines the key value mapping of headers to send.
		• **var** - defines the name of the variable to store the response in.
		• **mode** - defines what type of data to expect and how to parse it (supported: json).
		
		For example,
		
		`- type: request`
		`method: get`
		`url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/bank/branches`
		`var: branches`
		`mode: json`

assert-is	Assert that the type of variable from the response is **value**.	• **expression** - defines the variable to test.
		• **value** - defines the **type** we expect in response.
		
		For example,
		
		`- type: assert-is`
		`expression: branches`
		`value: array`

assert-exists	Assert that a particular variable specified in **expression** exists.	• **expression** - defines the variable to test.
		• **stoponfail** - If one of the test fails, the **stoponfail**attribute stops the execution of the remaining tests. For this, **stoponfail** must be set to **true**.
		
		For example,
		
		`- type: assert-exists`
		`expression: branches.[1]`
		`stoponfail: true`

assert-equals	Assert that a variable is equal to **value**.	• **expression** - defines the variable to test.
		• **value** - defines the **value** to compare against .
		
		For example,
		
		`- type: assert-equals`
		`expression: payload_response_header_Content-Type`
		`value: application/json`

assert-greater	Assert that the variable is greater than **value**.	• **expression** - defines the variable to test.
		• **value** - defines the **value** to compare against.
		
		For example,
		
		`- type: assert-greater`
		`expression: payload_temperature`
		`value: application/json`

assert-less	Assert that the variable is lesser than **value**.	• **expression** - defines the variable to test.
		• **value** - defines the **value** to compare against.
		
		For example,
		
		`- type: assert-less`
		`expression: payload_temperature`
		`value: application/json`

assert-in	Assert that the element identified by a given expression matches at least one item from a given list.	• **expression** - defines the variable to test.
		• **value** - defines an array of values to compare against.
		
		For example,
		
		`- type: assert-in`
		`expression: payload_response_status`
		`value: application/json`

Parameters	Description	Sub-attributes

`assert-matches`	Assert that the value of the element identified by a given expression matches a specified format.	• `expression` - defines the variable to test.
		• `value` - defines the regex value to compare against.
		
		For example,
		
		`- type: assert-matches`
		`expression: payload_response_header_Content-Type`
		`value: ^[a-z]+/json`

`assert-compares`	Assert that two elements are equivalent in some way.	• `expression1` - defines the path to the first element for comparison.
		• `expression2` - defines the path to the second element for comparison.
		
		For example,
		
		`- type: assert-compares`
		`expression1: branches.[0].type`
		`expression2: branches.[7].type`

`assert-contains`	Assert that the value of the element identified by a given expression contains a specific substring.	• `expression` - defines the variable to test.
		• `value` - defines the substring to compare against.
		
		For example,
		
		`- type: assert-contains`
		`expression: payload_response_header_Content-Type`
		`value: app`

`each`	Loop over the array referenced in the expression carrying out the sub-steps for each item found.	`expression` - defines the variable to test.
		
		For example,
		
		`- type: each`
		`expression: "analytics.events"`

For more information about assigning variables on a YAML configuration file, see Variable assignment in YAML configuration.

Variable assignment in YAML configuration

You can define variables within a YAML configuration file using the CLI tool. These variables are used to configure various aspects of the tool.

You can configure a YAML file in three ways:

- Using the same YAML file.
- Using environmental variables
- Using the external configuration YAML files.

Note: Variables must be enclosed in double curly braces. For example, `"{{configs.base_url}}"` or `"{{paths.basePath}}"`. These placeholders, in the format of `"{{...}}"`, will be replaced with their corresponding values from the source YAML file, environmental variables, or the external configuration file depending on the precedence hierarchy.
The precedence hierarchy for variable assignment is as follows:

1. External configuration YAML file (highest precedence)
2. Environmental variables
3. Configuration set in the same YAML file (lowest precedence)

If you configure a YAML file using all the three methods, the variables assigned using the external configuration YAML file is given priority.

1. Configuration set in the same YAML file

You can assign values to variables from within the same YAML file. For this, you must structure your YAML file in the following manner:

```
configs:
   basePath: ...paths:
       - basePath: ...
```

Here, the `configs` key is used to define variables such as `basePath` and `paths`. These variables can be utilized throughout the YAML file to configure the tool.

2. Environmental variables

You can override the values of variables configured in the YAML file using environmental variables. When an environment variable shares the same name as a variable defined in the YAML file, the value of the YAML variable is replaced by the corresponding environment variable. This provides a way to customize configurations externally.

For example, if a variable named `base_url` is defined in the YAML file as `"{{configs.base_url}}"`, and its value is replaced by an environment variable named `base_url`. The value of `base_url` in the YAML file is overridden by the value of the `base_url` environment variable.

3. External configuration YAML file

The CLI tool supports the highest level of configuration precedence through external configuration files. To use an external configuration file, you must specify its file path using the `--config-file` or `-c` flag when running the CLI tool. Any variables defined in the external configuration file will supersede those defined within the main YAML file or through the environmental variables.

For example, consider an external configuration file named `config.yaml` with the following content:

```
external_config:
   base_url: https://example.comapi_key: your-api-key
```

When you run the CLI tool with the `--config-file=config.yaml` flag, variables defined in the`config.yaml` file will replace the corresponding variables in the main YAML file or the environmental variables.

For more information about how the test is defined as a YAML file, see Test definition syntax.

Managing platform REST API keys

Create API keys associated with your user account, and then use them to authenticate when you call the platform REST APIs provided by API Connect.

About this task

The API Connect platform REST APIs require authentication for use. To call an API interactively, you can log into the toolkit CLI with an option to generate the API key. To call an API programmatically, you can generate an API key and then exchange it for an API Connect access token. This requirement applies only to the APIs that are provided by API Connect. You do not need the API key for calling APIs that you imported or created. The API key is intended for use with the IBMid and might not work with other user registries.

Procedure

1. Create an API key:
   a. Log in to API Connect.
   b. On the address bar, add `/apikeys` to the end of the URL (after the organization name).
      For example:

      ```
      https://manager.example.com/manager/myorg/apikeys
      ```

   c. On the API Keys page, click Add.
   d. In the "Create API Connect API Key" box, provide a title for the key (unique within your user account) and an optional description; then click Create.
2. Manage API Keys:
   a. Log in to API Connect.
   b. On the address bar, add `/apikeys` to the end of the URL.

Searching for items in API Manager

Use the search feature in IBM® API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.

Search is available from any page in API Manager and can be accessed by typing in the search field in the page banner, or by clicking Search in the navigation list.

Using Search

The Search feature is available on all pages of the API Manager. To start searching, just type a string into the Search field and press Enter. Whenever you press Enter in the Search field, the Search page displays with results and additional options. If you press Enter in an empty Search field, or click Search in the navigation list, then the Search page displays a list all items. If you type a string before pressing Enter, the page displays the results from the corresponding search.

Search results include all items that contain the search string in a text field. For example, suppose you searched for "Crafted". The results might look like the following image, with items that don't match the search string exactly, but do include the string in a text field such as a title or a name.

By default, the search returns all types of items that contain the string. Sort results on a particular column by clicking the column header.

When you find an item you want to work with, click the . . . actions menu next to the item's "Modified" date to see what you can do. The list of possible actions depends on the item's type.

You can limit the search results by selecting an item type from the list that displays before the results. Click More to view additional types. If you select a type from the More menu, it replaces a type in the list.

If you want to refine the search further, click Options. On the Options panel, click Catalogs to Search and select which catalogs to search (the default is all catalogs). You can select up to 5 catalogs for a limited search.

You can also choose whether only exact matches (the complete string) are included in results, or items containing similar strings are also included.

Working with API definitions

An API definition specifies the complete configuration for an API. You can create and configure your APIs either by using either the API Designer UI application, or by using the browser based API Manager UI.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
If you are working with APIs that conform to the OpenAPI Specification, note that API Connect uses the following terms:

- OpenAPI Specification:
  The OpenAPI Specification defines a standard, language-agnostic interface to RESTful APIs. API Connect supports version 2 and version 3 of the OpenAPI Specification.
- OpenAPI document:
  One or more files that describe an API conforming to the OpenAPI Specification.
- OpenAPI definition:
  The description of an API that conforms to the OpenAPI Specification. In API Connect, you can create the API definition by filling in settings and properties, or by importing an OpenAPI document that contains an API definition that you want to work with.

The following topics provide full details for creating and configuring your APIs.

- **Creating an API definition**
  You can create API definitions by using the API Designer or the command line interface in IBM® API Connect.
- **Editing an OpenAPI 2.0 API definition**
  IBM API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 2.0 specification.
- **Editing an OpenAPI 3.0 API definition**
  IBM API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 3.0 specification.
- **LDAP authentication**
  The Lightweight Directory Access Protocol (LDAP) is an internet protocol for accessing and maintaining distributed directory information services over a network. If you rely on LDAP to authenticate users for web applications, take a minute to review the contents of this topic before beginning.
- **Authentication URL user registry**
  You can use an Authentication URL user registry to specify a REST authentication service to manage user authentication, and to optionally provide additional metadata to be embedded in the token.
- **Activating an API**
  After you have created an API definition you can activate it to auto-publish your API document and make it available for testing.
- **Testing an API**
  Select a test tool and verify that your API works as you intended.
- **Staging an API**
  The graphical wizard provides an option that adds the API to a Product and stages the Product in a Catalog. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.
- **Publishing an API**
  The user interface provides an option that adds an API to a Product and publishes the Product in a Catalog. Publishing a draft Product makes the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.
- **Creating a new version of an API definition**
  You can create multiple versions of an API definition and edit the versions independently by using IBM API Connect.
- **Updating an API definition from a file**
  You can update an API by uploading a .yaml or .json file that contains the OpenAPI definition for the revised API.
- **Updating a SOAP API**
  You can update the configuration of an existing SOAP API either by uploading a WSDL file, or by uploading a .zip archive that contains a primary WSDL file together with other WSDL or XSD files that it references.
- **Renaming an API definition**
  By renaming an API definition, you change the `name` property that, together with the version, is used to uniquely identify the API definition in API Connect. The `name` property is initially generated automatically when you first create the API definition, based on the title, but you can later change it if desired.
- **Changing an API rate limit**
  When you activate an API to make it available for testing, as described in Activating an API, no rate limit is applied to restrict calls to the API. However, if desired, you can apply a rate limit to the API, or modify a previously applied rate limit. The rate limit defines the maximum number of calls allowed in a specified time period.
- **Adding an OpenAPI extension to an API**
  You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An OpenAPI extension is imported into a Catalog, or to a Space within a Catalog, then added to the API schema.
- **Downloading an API definition**
  You can download the details of your API definitions so that you can store them for later recovery.
- **Deleting an API definition**
  You can delete an API definition that is no longer required.
- **Using an options file when importing a WSDL service**
  When you create an API definition, or add a target WSDL service to an API definition, by importing a .zip file, you can specify additional directives by including an options file in the .zip file.
- **Variable references in API Connect**
  In API Connect you can reference different variables in your API definition.
- **API properties**
  Properties are used by the gateway to control behavior of certain policies. Typically, you provide properties, but the policy can also provide properties settings.
- **API policies and logic constructs**
  Policies and logic constructs are pieces of configuration that control a specific aspect of processing in the Gateway server during the handling of an API invocation at run time.
- **Tags in API Connect**
  There are a number of forms of tagging in IBM API Connect.

Creating an API definition

You can create API definitions by using the API Designer or the command line interface in IBM® API Connect.

Before you begin

An API is a set of functions that provide some business or technical capability and can be called by applications by using a defined protocol. Applications are typically mobile or web applications, and they use the HTTP protocol. An API definition is composed of paths, and can be one of the following types:

REST API definition
: A REST API is a defined set of interactions that uses the HTTP protocol, typically by using JSON or XML as the data format that is exchanged. For example, a data request might use an HTTP GET method, and a data record might use an HTTP POST method. The choice of data format depends on the type of application that is calling the API. JSON is commonly used for web pages or mobile applications that present a user interface (by using JavaScript or HTML), whereas XML is often used for machine-to-machine scenarios.

  You can create REST APIs by using LoopBack® functions to create models and data sources. These are then used by your REST API definition and exposed to your users.

  Alternatively, you can expose and secure your existing APIs by using a Proxy or Invoke policy.

  In either case, you can configure your API definition either by using the API Designer, or by writing an OpenAPI definition file and publishing it using either API Designer or the command line interface.

SOAP API
: You can create SOAP API definitions that are based on an existing Web Services Description Language (WSDL) file. You can use this facility to benefit from the capabilities that are provided by API Connect, which include analytics and mapping between variables. You can also expose the API by using the Developer Portal for any existing SOAP services in your organization, including any SOAP services that are part of a service-oriented architecture (SOA) or Enterprise Service Bus (ESB) infrastructure.

  You can create SOAP API definitions through either the command line interface, or through API Designer.

About this task

You can create definitions for REST APIs or SOAP APIs.

- In the API Designer, you can add a REST API definition either by composing the API definition, and its operations, from scratch, or by importing an OpenAPI definition file. You can also use the tooling to quickly create a proxy API that calls an existing endpoint.
- If you have an existing SOAP service that you want to expose more widely, you can add a SOAP API to IBM® API Connect. You can use the Developer Portal to publicize the SOAP service to the developers. If a developer wants to use the SOAP API, you can use API Connect to manage their sign-up and access to the service, and track the usage of that API.

Procedure

1. Quickly create an API definition by using one of the following wizards:
   - Creating a REST proxy API from a target service
   - Creating a REST proxy API from an existing OpenAPI service
   - Creating a new REST OpenAPI definition
   - Creating a REST proxy API from an existing WSDL service
   - Adding a REST API by importing an OpenAPI definition file
   - Creating a SOAP proxy API from an existing WSDL service
2. After you create your API definition, you can define it in more detail by following the instructions in Editing an OpenAPI 2.0 API definition.

- **Creating a REST proxy API from a target service**
  If you have an existing REST service that you want to expose through an IBM API Connect API definition, you can create a proxy API and specify the target endpoint by using the API Designer.
- **Creating a REST proxy API from an existing OpenAPI service**
  If you have an existing OpenAPI described target service that you want to expose through an IBM API Connect API definition, you can create a proxy API and specify the target endpoint.
- **Creating a REST proxy API from an existing WSDL service**
  If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect REST proxy API that calls that SOAP service.
- **Creating a SOAP proxy API from an existing WSDL service**
  If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect SOAP proxy API that calls that SOAP service.
- **Creating a new REST OpenAPI definition**
  You can create and edit draft REST API definitions by using the API Designer or the API Designer user interface in IBM API Connect.
- **Adding a REST API by importing an OpenAPI definition file**
  You can use an OpenAPI definition file to import a REST API into IBM API Connect.
- **Adding a SOAP API by importing a zip file**
  You can use an OpenAPI definition file to import a SOAP API into IBM API Connect.
- **Creating a GraphQL proxy API**
  GraphQL is a query language for APIs that gives an application client greater control over what data it retrieves in an API request when compared with a REST API request. IBM API Connect enables you to create a GraphQL API proxy definition that proxies a backend GraphQL server, and to define rate limiting controls that reflect the amount of data that is returned from the server by a request to the GraphQL API.
- **Importing an API from the CP4I asset repository**
  If you are using IBM API Connect with IBM Cloud Pak for Integration (CP4I), you can import an API definition from the CP4I asset repository.

Related concepts

- Using the developer toolkit command-line tool

Creating a REST proxy API from a target service

If you have an existing REST service that you want to expose through an IBM® API Connect API definition, you can create a proxy API and specify the target endpoint by using the API Designer.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

Create proxy REST APIs in minutes by using the API Designer tool to create an API from target service. This procedure creates a REST proxy that routes all traffic to a target REST API or service endpoint.

Procedure

To compose a proxy API from a target service, complete the following steps.

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select From target service.
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **info.version** property of the API's OpenAPI definition. The **version.release.modification** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
5. Specify the existing REST API endpoint that you want to call in the Target Service URL field.
6. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **X-IBM-Client-Id** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
     - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

     For information about security options in IBM API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **cors** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **cors** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **request.attributes.isCORSPreflight** flag is set to **true**.
     - For all preflight requests, the **security** and **client-identification** preflow policies are always skipped, whether CORS is enabled or not enabled.
7. Click Next to create your API definition.
   The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
8. Select one of the following options:
   - To further configure your API, click Edit API. For details, see Editing an API definition.
   - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

You created a proxy API from an existing target service.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

Related tasks

- Defining Paths for a REST API

Related reference

- API and Product definition template examples

Creating a REST proxy API from an existing OpenAPI service

If you have an existing OpenAPI described target service that you want to expose through an IBM® API Connect API definition, you can create a proxy API and specify the target endpoint.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **Api-Drafts:Edit**, **Settings:View**, and **App:View** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

Create proxy REST APIs in minutes by using the API Designer tool to create an API from existing OpenAPI service. This procedure creates a REST proxy that routes all traffic to a target REST API or service endpoint.

Procedure

To compose a proxy API from an existing OpenAPI service, complete the following steps.

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select From existing OpenAPI service, and click Next.
4. To upload the service information from an OpenAPI file, you can either drag and drop your file, or browse and select the file that you want to use. Valid file types are YAML, YML, etc.
   - Click Browse and browse for the OpenAPI file.
   - Drag and drop the file into the active area.

   After you upload the file, the OpenAPI specification is evaluated and a message displays whether it is valid.
5. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **info.version** property of the API's OpenAPI definition. The **version.release.modification** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
6. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **X-IBM-Client-Id** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
     - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

     For information about security options in IBM API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **cors** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **cors** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **request.attributes.isCORSPreflight** flag is set to **true**.
     - For all preflight requests, the **security** and **client-identification** preflow policies are always skipped, whether CORS is enabled or not enabled.
7. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:
     - Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
     - Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.
   - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
   - To use the Activate API option, you must be assigned a role that has the **Product:Manage** and **Subscription:Manage** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.
8. Click Next to create your API definition.
   The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.

   If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.

9. Select one of the following options:
   - To further configure your API, click Edit API. For details, see Editing an API definition.
   - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

You created a proxy API from an existing OpenAPI service.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

Related tasks

- Defining Paths for a REST API

Related reference

- API and Product definition template examples

Creating a REST proxy API from an existing WSDL service

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect REST proxy API that calls that SOAP service.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`**, **`Settings:View`**, and **`App:View`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

If the WSDL file is a stand-alone file with no external dependencies, you can load the .wsdl file from a directory to create the REST API definition.

If the WSDL file references other WSDL files or references XSD files containing XML schema definitions, you must create a .zip archive of the WSDL file and its dependent documents, and then load the .zip file from a directory to add the REST API definition.

Procedure

To add a REST API definition by loading a WSDL file, complete the following steps:

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. On the Add API pane, select From existing WSDL service (REST Proxy). Click Next.
   The Create API from Existing WSDL Service (REST Proxy) window opens.
4. To upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, you can either drag and drop your file, or browse and select the file that you want to use.
   After you upload the file, the WSDL is evaluated and a message displays whether the WSDL is valid.

   If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see Using an options file when importing a WSDL service.

5. Click Next. At the Select Service panel, select a WSDL service from the imported file.
6. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **`info.version`** property of the API's OpenAPI definition. The **`version.release.modification`** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
7. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **`X-IBM-Client-Id`** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
     - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

     For information about security options in IBM® API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **`cors`** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **`cors`** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **`request.attributes.isCORSPreflight`** flag is set to **`true`**.
     - For all preflight requests, the **`security`** and **`client-identification`** preflow policies are always skipped, whether CORS is enabled or not enabled.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:

- Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
- Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.

- You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
- To use the Activate API option, you must be assigned a role that has the **`Product:Manage`** and **`Subscription:Manage`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

9. Click Next to create your API definition.
   The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.

   If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.

10. Select one of the following options:
    - To further configure your API, click Edit API. For details, see Editing an API definition.
    - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

You created a REST API definition by using an existing WSDL file.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

You can also manage the Product lifecycle, and control who can see and subscribe to your Product, by opening the Sandbox Catalog associated with your API in the Manage page of the API Designer UI.

Related concepts

- Developing your APIs and applications

Related reference

- API and Product definition template examples

Related information

- IBM API Connect overview

Creating a SOAP proxy API from an existing WSDL service

If you have a SOAP service defined in a WSDL file, you can use the WSDL file to create an API Connect SOAP proxy API that calls that SOAP service.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`**, **`Settings:View`**, and **`App:View`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

Note: By default, the SOAP proxy API that is created will have the basepath set to the WSDL service name. You can edit the SOAP proxy API to change this basepath.
You can add a SOAP API to expose an existing SOAP service by supplying the WSDL file that defines that existing service in one of the following ways:

- If the WSDL file is a stand-alone file with no external dependencies, you can load the .wsdl file from a directory to create the SOAP API definition.
- You can provide one or more WSDL files in a single .zip file that contains the WSDL files and any necessary schemas.
  If the WSDL file references other WSDL files or references XSD files containing XML schema definitions, you must create a .zip archive of the WSDL file and its dependent documents, and then load the .zip file from a directory to add the SOAP API definition.

- If you are using API Designer with existing APIs that were created from a WSDL or WSDL.zip file in Version 5 API Designer, make sure that the following requirements are met:
  - The WSDL or WSDL.zip file must be stored in the same directory as the API's YAML file.
  - In the API's YAML file, the **`x-ibm-configuration`** --> **`wsdl-definition`** --> **`wsdl`** property must point directly to the WSDL or WSDL.zip file with no path specified (the files are located in the same directory).

The service must support Web Services Basic Profile Version 1.1 - Second Edition.

Procedure

To add a SOAP API definition by loading a WSDL file, complete the following steps:

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. Select From existing WSDL service (SOAP Proxy). Click Next.
   The Create API from Existing WSDL Service (SOAP Proxy) window opens.
4. To upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, you can either drag and drop your file, or browse and select the file that you want to use.
   After you upload the file, the WSDL is evaluated and a message displays whether the WSDL is valid.

   If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see Using an options file when importing a WSDL service.

5. Click Next. At the Select Service panel, select a WSDL service from the imported file.
6. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **`info.version`** property of the API's OpenAPI definition. The **`version.release.modification`** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
7. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **`X-IBM-Client-Id`** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
     - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

     For information about security options in IBM® API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **`cors`** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **`cors`** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **`request.attributes.isCORSPreflight`** flag is set to **`true`**.
     - For all preflight requests, the **`security`** and **`client-identification`** preflow policies are always skipped, whether CORS is enabled or not enabled.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:
     - Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
     - Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.
   - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
   - To use the Activate API option, you must be assigned a role that has the **`Product:Manage`** and **`Subscription:Manage`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.
9. Click Next to create your API definition.
   The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.

   If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.

10. Select one of the following options:
    - To further configure your API, click Edit API. For details, see Editing an API definition.
    - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

You created a SOAP API definition by using an existing WSDL file.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

You can also manage the Product lifecycle, and control who can see and subscribe to your Product, by opening the Sandbox Catalog associated with your API in the Manage page of the API Designer UI.

Related concepts

- Developing your APIs and applications

Related reference

- API and Product definition template examples

Related information

- IBM API Connect overview

Creating a new REST OpenAPI definition

You can create and edit draft REST API definitions by using the API Designer or the API Designer user interface in IBM® API Connect.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task you must be assigned a role that has the **`Api-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Procedure

To create a new REST OpenAPI definition, complete the following steps.

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. Select New OpenAPI .
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **`info.version`** property of the API's OpenAPI definition. The **`version.release.modification`** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
5. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **`X-IBM-Client-Id`** parameter to be included in the request header for the API. If selected, you can then select whether to limit the API calls on a per key (per Client ID) basis:
     - Limit API calls on a per key basis - If selected, you must configure the rate limit that you require. Rate limits control the maximum number of calls allowed within a time period (hour, minute, month or day). For example, 100 calls per hour.

     For information about security options in IBM API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **`cors`** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **`cors`** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **`request.attributes.isCORSPreflight`** flag is set to **`true`**.
     - For all preflight requests, the **`security`** and **`client-identification`** preflow policies are always skipped, whether CORS is enabled or not enabled.
6. Click Next to create your API definition.
   The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.
7. Select one of the following options:
   - To further configure your API, click Edit API. For details, see Editing an API definition.
   - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

You successfully created a REST API definition. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

Related tasks

- Defining Paths for a REST API
- Configuring API security (OpenAPI 2.0)
- Configuring API security (OpenAPI 3.0)

Related reference

- API and Product definition template examples

Adding a REST API by importing an OpenAPI definition file

You can use an OpenAPI definition file to import a REST API into IBM® API Connect.

Before you begin

Your file must conform to version 2.0 or version 3.0 of the OpenAPI specification. The format of the file can be JSON or YAML.

Note: Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "`(?`" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
    definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the `Api-Drafts:Edit`, `Settings:View`, and `App:View` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

If you are using the API Designer UI and you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, use the import mechanism described here rather than copying the file into your local API Designer directory; the import operation adds API Connect specific sections that are required by API Designer.

Procedure

To add a REST API by importing an OpenAPI file, complete the following steps:

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. In the Import section, select Existing OpenAPI, then click Next.
4. Choose one of the following methods for importing the file:
   - Drag and drop a file
     Open another window, select a file. Drag the file to the Select a file box, and release the file.
   - Browse for a file
     Click the Select a file box. Locate the file and select it.
   - Specify the URL of a file
     In the Or specify a file URL field, type or paste a URL for the file. Optionally add a User name and Password if they are required for accessing the file.

   The following file types are supported if they contain a valid OpenAPI definition: .json, .yml, and .yaml.
5. Click Next to import the selected file.
   The wizard checks the validity of the YAML, and displays a message indicating successful validation.

   Notes:
   - If you are using the API Manager UI, the import operation fails if the file defines an API that has the same name and version as an existing API definition. However, if you are using the API Designer UI, API definitions are uniquely identified by the file name in your local file system; therefore, if you import two different files that define the same API name and version, two API definitions with the same name and version are created in API Connect, with no error.
   - Any validation error messages are displayed in English only, and are not translated.
6. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:
     - Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the

API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
    - Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.
  - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
  - To use the Activate API option, you must be assigned a role that has the **`Product:Manage`** and **`Subscription:Manage`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.
7. Click Next. The Import API Summary panel indicates that the YAML file is loaded and valid.
   If you selected Activate API, the wizard populates an API Endpoint URL, and displays a Client ID and Client Secret that you can use.
8. Select one of the following options:
   - To further configure your API, click Edit API. For details, see Editing an API definition.
   - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

When the API definition has been imported, it is shown in the list of API definitions in the Develop page. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

Related information

- IBM API Connect overview
- Managing your APIs
- OpenAPI Specification
- What is OpenAPI?
- API and Product definition template examples

Adding a SOAP API by importing a zip file

You can use an OpenAPI definition file to import a SOAP API into IBM® API Connect.

Before you begin

The format of the file must be .zip, and the file must contain the WSDL definition and the YAML file that defines the API. The YAML file must conform to version 2.0 of the OpenAPI specification.

Note: Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "**`(?`**" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
    definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`**, **`Settings:View`**, and **`App:View`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

If you are using the API Designer UI and you want to use an OpenAPI definition file from elsewhere, downloaded from an external website for example, rather than created by using API Connect, use the import mechanism that is described here rather than copying the file into your local API Designer directory; the import operation adds API Connect specific sections that are required by API Designer.

For more information on how to create a .zip file, see Downloading an API definition.

Procedure

To add a SOAP API by importing a .zip file, complete the following steps:

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.

3. In the Import section, select Existing OpenAPI, then click Next.
4. To import a file from your local file system, you can either drag and drop the file or click the link to select the file from your local file system.
   The wizard checks the validity of the WSDL and YAML, and displays a message that indicates a successful validation.
   Note:
   - If you are using the API Manager UI, the import operation fails if the file defines an API that has the same name and version as an existing API definition. However, if you are using the API Designer UI, API definitions are uniquely identified by the file name in your local file system; therefore, if you import two different files that define the same API name and version, two API definitions with the same name and version are created in API Connect, with no error.
   - Any validation error messages are displayed in English only, and are not translated.
5. Click Next.
6. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:
     - Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
     - Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.
   - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
   - To use the Activate API option, you must be assigned a role that has the **`Product:Manage`** and **`Subscription:Manage`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.
7. Click Next. The Import API Summary panel indicates that the file is loaded and valid.
   If you selected Activate API, the wizard populates an API Endpoint URL, and displays a Client ID and Client Secret that you can use.
8. Select one of the following options:
   - To further configure your API, click Edit API. For details, see Editing an API definition.
   - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Results

When the API definition has been imported, it is shown in the list of API definitions in the Develop page. For API Designer, the specifications for the APIs and Products are stored in the directory that you specified when you logged in. For API Manager, the specifications for the APIs and Products are stored on the management server.

What to do next

APIs are made available to application developers by including them in a Product, and then publishing that Product to a Catalog. For more information, see Working with Products and Working with Catalogs.

Related information

- IBM API Connect overview
- Managing your APIs
- OpenAPI Specification
- What is OpenAPI?
- API and Product definition template examples

API Gateway only

Creating a GraphQL proxy API

GraphQL is a query language for APIs that gives an application client greater control over what data it retrieves in an API request when compared with a REST API request. IBM® API Connect enables you to create a GraphQL API proxy definition that proxies a backend GraphQL server, and to define rate limiting controls that reflect the amount of data that is returned from the server by a request to the GraphQL API.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`**, **`Settings:View`**, and **`App:View`** permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

GraphQL provides the following particular advantages over REST APIs:

- The application client can request only the data that it needs. For example, when retrieving bank account records, request only the account number and current balance for each account, but not the customer name and address details. With a REST API request, either the backend REST service must provide separate endpoints for different data subsets, or the application client must retrieve the complete records and then discard the unwanted data.
- The application client can retrieve multiple related resources in a single request. For example, a customer's bank account record might include an array that references other finance products that the customer holds. If an application client wants to retrieve the bank account details for a specific customer, and details of the other finance products for that customer, then with a REST API the client would first retrieve the bank account details, then make separate requests for each of the other products. A GraphQL API can be designed to allow the client to retrieve all this information in a single request.

However, this flexibility presents rate limiting challenges, because two seemingly very similar requests might return vastly different amounts of data, and what might have required multiple REST API requests, each counting towards the rate limit, might require only a single GraphQL API request. It is important therefore that rate limiting controls are imposed that reflect the amount of data that is retrieved. API Connect extends the GraphQL standard by providing, in a GraphQL API definition, the ability to configure a range of settings that are used to calculate the complexity of a GraphQL request and an associated cost that counts towards the rate limit.

Procedure

To create a GraphQL API proxy definition, complete the following steps:

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0; this API type does not support OpenAPI 3.0.
3. Select From existing GraphQL service (GraphQL proxy).
4. Click Next. Specify the API summary in the Info section. You can fine-tune the API after it is created.
   - The Title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The Name is entered automatically. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - The Version corresponds to the value of the **`info.version`** property of the API's OpenAPI definition. The **`version.release.modification`** version numbering scheme is recommended; for example `1.0.0`.
   - The Base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The optional Description helps to identify the API.
5. In the GraphQL Server section, specify the following values:
   - GraphQL server URL: this is the URL that is used for requests to the backend GraphQL API. The GraphQL schema is imported automatically from the specified URL.
   - Schema name: the name that you enter here is used to identify the schema definition that is created from the imported GraphQL schema. The field is pre-populated with host name segment of the value entered in the GraphQL server URL field, but you can change it.
6. Click Next. If there are any warnings indicated relating to the imported schema, you can review them now by clicking View.
   You can deal with these warnings as appropriate after you have created your GraphQL proxy API.
7. In the Operations and Paths section, select the operations and paths that you want to include.
   The operation path of **`/graphql`** is already selected and you cannot change this setting; this is the path for application client requests to API Connect that invoke the GraphQL API.

   For details of the mechanisms supported for sending a request to the **`/graphql/cost`** endpoint, see Request mechanisms supported by GraphQL endpoints.

   You can also select the operation path of graphql/cost; this path enables application clients to obtain details of the cost of a request to the GraphQL API before making the actual request. In a production environment, in particular, you should consider carefully the resource implications of making this path available. For full details regarding the enablement of the cost endpoint, see Enabling the cost endpoint for a GraphQL API.

   To form the full API paths, these path segments are appended to the **`basepath`** that you specified in step 4.

   To allow application clients to send introspection requests for this GraphQL proxy API, select Support default introspection. For full details, see Supporting introspection for a GraphQL API.

   To allow users to test the GraphQL proxy API from a GraphiQL editor, select Enable GraphiQL editor. For full details, see Enabling the GraphiQL editor for a GraphQL API.

   Note: The graphql/cost, Support default introspection, and Enable GraphiQL editor options each automatically generate policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required.
8. Optionally, you can replace the GraphQL schema that was imported from the GraphQL server URL by uploading a schema from your local file system. Click Replace; you can then either drag and drop your file, or click where indicated to select the file from your local file system. Click Upload when done.
   Note:
   - The schema must be written in the GraphQL Schema Definition Language (SDL). For more information, see Type language in the GraphQL documentation at https://graphql.org/.
   - If you upload a schema from a local file having previously uploaded a schema from a URL, the schema URL setting that is stored in the OpenAPI source for the API, and subsequently displayed in the user interface, retains the previously specified URL value rather than indicating a file location.
9. Click Next. In the Secure section, configure the API security that you require.
   - Secure using Client ID - Select this option to require an Application to provide a Client ID (API Key). This causes the **`X-IBM-Client-Id`** parameter to be included in the request header for the API. For information about security options in IBM API Connect, see Configuring API security.
   - CORS - Select this option to enable cross-origin resource sharing (CORS) support for your API. This allows your API to be accessed from another domain.
     Note:
     - CORS support is available only on the DataPower® API Gateway.
     - When CORS is enabled, the API Gateway runs the **`cors`** preflow policy to handle all CORS requests that are made to the API.
     - When CORS is enabled and a preflight request is received, only the following API actions are performed:
       - The **`cors`** preflow policy configures the appropriate response headers.
       - The response headers are set.
     - When a preflight request is received, the **`request.attributes.isCORSPreflight`** flag is set to **`true`**.
     - For all preflight requests, the **`security`** and **`client-identification`** preflow policies are always skipped, whether CORS is enabled or not enabled.
10. Optional: Select Activate API if you want to immediately use the API for further development and testing.
    Note:
    - When you select the Activate API option, API Connect automatically completes the following actions:
      - Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
      - Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.
    - You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and

configuring catalogs.
    - To use the Activate API option, you must be assigned a role that has the `Product:Manage` and `Subscription:Manage` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.
11. Click Next to create your API definition.
    The Summary panel displays messages as the definition is created, and the selected security options are enforced.

    If you selected Activate API, the wizard populates an API Endpoint URL that you can use in testing. If you have also selected Secure using Client ID, the wizard displays a Client ID and Client Secret you can use.

12. Click Next to create your API definition.
    The Summary panel displays messages as the definition is created, and the selected security options and rate limits are enforced.

13. Select one of the following options:
    - To further configure your API, click Edit API. For details, see Editing an API definition.
    - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

What to do next

Use the GraphQL schema editor to review, and deal with, any warnings associated with the schema, and to configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit; for details, see Using the GraphQL schema editor.

- **Using the GraphQL schema editor**
  The GraphQL schema editor allows you to a configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit.
- **Supporting introspection for a GraphQL API**
  If you select the Support default introspection option when creating a GraphQL proxy API, client applications can send GraphQL introspection requests to the API.
- **Enabling the GraphiQL editor for a GraphQL API**
  If you select the Enable GraphiQL editor option when creating a GraphQL proxy API, users can test the API by using a GraphiQL editor in a web browser; API Connect responds to the initial GraphiQL editor request with an HTML page that enables testing from the GraphiQL editor.
- **Enabling the cost endpoint for a GraphQL API**
  If you enable the `/graphql/cost` endpoint when creating a GraphQL proxy API, a calling application can obtain details of the cost of executing a request on the backend GraphQL server before making the actual backend request.
- **Securing a GraphQL API by using a client ID**
  If you select the Secure using Client ID option when creating a GraphQL proxy API, API Connect automatically adds Rate limit policies to the API assembly.
- **Request mechanisms supported by GraphQL endpoints**
  API Connect supports various mechanisms for sending requests to GraphQL proxy API endpoints.
- **GraphQL custom schema directives and scalar types**
  The API Connect DataPower API Gateway extends the GraphQL specification by providing additional schema directives and custom scalar types.
- **@remove directive**
  The `@remove` GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context.
- **GraphQL limitations**
  This page details the limitations in the API Connect GraphQL implementation for the current release.

API Gateway only

Using the GraphQL schema editor

The GraphQL schema editor allows you to a configure a range of settings that are used to calculate the complexity of a GraphQL request, and an associated cost that counts towards the rate limit.

Introduction

When the GraphQL schema editor opens, all the types defined in the GraphQL schema are listed. You can expand a type to display all the fields that are defined for that type.

For each type, you can specify the following setting:

Type weight
: The weighting factor that you want to apply to the type for use when calculating the type cost for a request to the GraphQL API. In general, the weighting factor for a type will reflect the resource implications for a request that retrieves values of that type; for example:

  - A complex object might be weighted more heavily than a String.
  - Values of one type might be retrieved from a different backend database to values of another type, and accessing one database requires more CPU resource, so types associated with this database are weighted more heavily.

For each field, you can specify the following settings:

Field weight
: The weighting factor that you want to apply to the field for use when calculating the field cost of a request to the GraphQL API.

Assumed size
: The size limit that the backend server imposes on the number of values of this field that are returned, in a list, for each separate retrieval of the field in an API request. If an assumed size has been defined for this field in the imported GraphQL schema, its value is displayed. You an enter a value, or change the current value, and the schema definition stored in API Connect is updated accordingly. You must ensure that the specified value corresponds with the implementation or configuration on the backend server so that API Connect can apply rate limiting correctly. The assumed size is intended to be configured for fields that return lists of values, to tell the cost analysis how many values to expect.

Slicing arguments
: A comma separated list of the arguments that are used by the backend server to limit the number of values of this field that are returned, in a list, for each separate retrieval of the field in an API request. These must correspond with the implementation or configuration on the backend server. For example, there might be slicing arguments of **`first`** and **`last`** that are used to specify a range of objects to be returned by an API request.

Sized fields
: In some cases, a field with a configured Assumed size or Slicing arguments setting returns an object, a field of which returns the values whose number is to be sized. This is common in the connections pattern for pagination, where fields with slicing arguments **`first`** or **`last`** return a **`connections`** object, that contains the sized list in an **`edges`** field. The Sized fields option allows the cost analysis to consider these indirect relationships between assumed sizes or slicing arguments and the sized list of values. Supply a comma separated list of field names.

You access the GraphQL schema editor in either of the following ways:

- Immediately after creating a new GraphQL proxy API , by clicking Edit API, then clicking the GraphQL Schema tab. For GraphQL proxy API creation details, see Creating a GraphQL proxy API.
- By modifying the configuration of an existing API; complete the following steps:
  1. In the navigation pane, click Develop, then select the APIs tab.
  2. Alongside the GraphQL API definition that you want to work with, click the options icon and then click Edit API.
  3. Click the GraphQL Schema tab.

Modifying a setting for a single type or field

For a numerical setting, click the setting alongside the required type or field, then enter the required value, or use the stepper control. For a setting that has a textual value, enter the value directly. You might first want to filter the display to show only a relevant subset of types and fields; see Filtering the display.

Modifying settings for multiple types and fields

To modify settings for multiple types and fields in a single operation, complete the following steps:

1. Select the check boxes alongside the types and fields whose settings you want to modify.
2. Click Apply analysis configuration.
3. On the Type tab, specify the settings that you want to apply to the selected types.
4. On the Field tab, specify the settings that you want to apply to the selected fields.
5. Click Apply to apply the settings to the selected types and fields.
6. When you have made all the required modifications to the selected types and fields, click Done; this actions clears all check box settings.

To to restore any type or field settings to the initial default settings after having applied an analysis configuration, select the check boxes alongside the required types and fields, then click Remove analysis configuration.

You might first want to filter the display to show only a relevant subset of types and fields; see Filtering the display.

Filtering the display

To display only those type and fields that contain a specified string, enter the required filter string in the Search schema field, then press Enter.

If a type matches the filter, the type and all its fields are displayed. If a field matches the filter but not its parent type, the parent type is displayed together with all matching fields. You can enter multiple filter strings; the items displayed are those that contain all strings.

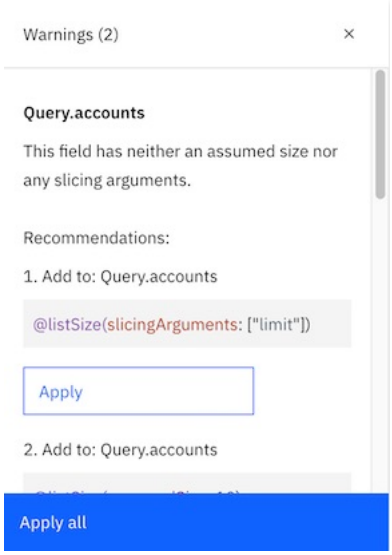
You can now use the operations described in Modifying a setting for a single type or field and Modifying settings for multiple types and fields to update the settings for the displayed types and fields.

To remove a specific filter string from the search, click the Close icon for that filter string. To restore the full type and field display by removing all filter strings, click the Clear icon at the end of the search field.

GraphQL schema warnings

Where applicable, API Connect will recommend updates to the analysis configuration to protect the backend server from excessive resource utilization, and to ensure correct cost estimation for requests. These recommendations are displayed as warnings in the GraphQL schema editor.

If a type contains one or more fields with warnings, this is indicated by a warning count icon 2 alongside the type. You can click the warning count icon to see the warnings for that type; the Warnings window opens at the relevant warnings, for example:



To see the fields that have warnings, expand the type; a warning icon is displayed alongside fields with warnings. To see the details of a warning for a field, click the warning icon alongside the field

If you click Accept under any warning in the Warnings window, the recommended update is applied to the schema automatically.

To apply the suggested updates for all warnings in a single operation, click Apply all in the Warnings window, then click Apply to confirm.

You can filter the display to include only items with warnings; click the Menu icon, then select Show warnings only.

Applying @remove directives to types and fields in the GraphQL schema

The **`@remove`** GraphQL schema directive specifies conditions for removing items from validation or introspection for each transaction based on values in the API context. For example, you can prevent specific Plans from using certain types or fields.

For more details on the **`@remove`** directive, including examples, see rapic_graphql_extensions.html#reference_wrt_bk1__5mb__at_remove.

To apply an **`@remove`** directive to an item in your GraphQL schema, complete the following steps:

1. Click the settings icon in the Show/hide column alongside the required item. The Show/hide settings window opens.
2. To remove the item depending on a boolean condition, enter the boolean expression in the Custom expression field; the following directive will be added to the item in the schema:

   `@remove(if: [`*`expression`*`])`

   To remove the item unconditionally, change the Show in schema slider control to Hide in schema; the following directive will be added to the item in the schema:

   `@remove(if: ["true"])`
3. Click Next. A summary window opens. If the removal is permitted, any related items that will also have the **`@remove`** directive applied are listed; for example, if you apply an **`@remove`** directive to a type, and that type is referenced as the data type of a field on another type, the **`@remove`** directive is also applied to that field. If the removal is not permitted, an explanation is displayed.
4. If the removal is permitted, click Done to apply the **`@remove`** directive. If the removal is not permitted, either click Cancel to close the window or click Back and amend your settings.
5. To see the updates to the GraphQL JSON schema, click View source.

You can use the settings icon to modify or delete previously applied **`@remove`** directives. To remove a conditional **`@remove`** directive, delete the expression from the Custom expression field. To remove an unconditional **`@remove`** directive, change the Hide in schema slider control to Show in schema.

Viewing the GraphQL schema

To view the GraphQL JSON schema that was previously uploaded, click View source. Note that the schema includes **`directive`** statements and attribute values, added by API Connect, that reflect the configuration settings.

Replacing the GraphQL schema

To replace the GraphQL schema, complete the following steps:

1. Click Replace.
2. Specify the replacement schema:
   - To replace the schema by introspecting a URL, complete the following steps:
     - Select Introspect from URL.
     - Enter the URL in the GraphQL server URL field, then click Introspect.

     Note: If you replace the schema from a URL that is different to the one specified as the backend GraphQL server, in the GraphQL server URL field, when the GraphQL proxy API was created, the schema itself is replaced but the backend URL remains unchanged. You must correspondingly modify the URL setting on the **`invoke`** policy in the API assembly of your GraphQL proxy API.
   - To replace the schema by uploading a file, complete the following steps:
     - Select Upload schema definition language (SDL) file.

- Either drag and drop the file or click the link to select the file from your local file system.

Note:

- The schema must be written in the GraphQL Schema Definition Language (SDL). For more information, see Type language in the GraphQL documentation at https://graphql.org/.
- If you upload a schema from a local file having previously uploaded a schema from a URL, the schema URL setting that is stored in the OpenAPI source for the API, and subsequently displayed in the user interface, retains the previously specified URL value rather than indicating a file location.

All the changes between the replacement schema and the current schema are listed.

3. Update the schema as required:
   - To have all new types and fields shown by default, select New schema for the Update schema defaults giving preference to setting. You can then choose to hide specific items by selecting Hide in the Action column alongside an item. Hiding an item causes an `@remove` directive to be added to the item in the schema source, as detailed in Applying @remove directives to types and fields in the GraphQL schema.
   - To have all new types and fields hidden by default, select Old schema for the Update schema defaults giving preference to setting. You can then choose to show specific items by selecting Show in the Action column alongside an item.

   Note: If the current schema contains items that are not in the replacement schema, these items are deleted. Such deletions are shown in the change listing with the Accept setting displayed in the Action column; you cannot change this setting.
4. Click Next. A change summary page is displayed. If any removals are not permitted, an explanation is displayed, and you cannot complete the schema replacement; either click Cancel to close the window or click Back and amend your settings..
5. Click Submit to complete the schema replacement.

Downloading the GraphQL schema

To download the GraphQL schema to a local file, click the Download. Note that the downloaded schema includes all additional settings that reflect your configuration updates.

Related reference

- GraphQL custom schema directives and scalar types

API Gateway only

Supporting introspection for a GraphQL API

If you select the Support default introspection option when creating a GraphQL proxy API, client applications can send GraphQL introspection requests to the API.

Introspection allows client applications to request details of all the types and fields that are defined in a GraphQL schema. When you create a GraphQL proxy API, introspection requests sent to that API from client applications are handled by the gateway. The response reflects any modifications you have made to the schema by using the GraphQL schema editor.

When you select the Support default introspection option, API Connect creates the introspection endpoint. A GraphiQL editor always attempts to introspect a GraphQL endpoint, so disabling this option significantly reduces the test capability for GraphiQL editor users. This option also automatically generates policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required; see the API assembly policies for introspection support section for details. Disabling this option also limits the capability of the Developer Portal to test the GraphQL proxy API.

On the other hand, providing introspection support can have resource implications, and you might choose to communicate the schema information in another way. However, note that the GraphiQL editor does not allow custom introspection queries to populate the document explorer. For more information on introspection, see Introspection in the GraphQL documentation at https://graphql.org/.

API assembly policies for introspection support

This section details the API assembly policy configuration that is generated automatically if the Support default introspection option is selected when a GraphQL proxy API is created. If the option is not selected at creation time, and you later decide that you want to enable it, you must add it manually to the API assembly.

A `case` is added to a `switch` policy with the following condition:

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'standard-introspection')
```

If the condition is true, the following policies are executed in sequence, with property settings as indicated:

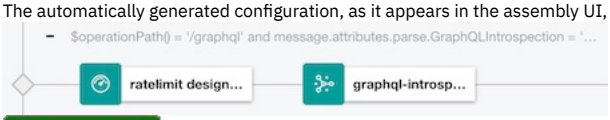
Rate limit

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

GraphQL introspect

Property	Value
Input	message
Output	No value, defaults to input

The automatically generated configuration, as it appears in the assembly UI, is as follows:



API Gateway only

Enabling the GraphiQL editor for a GraphQL API

If you select the Enable GraphiQL editor option when creating a GraphQL proxy API, users can test the API by using a GraphiQL editor in a web browser; API Connect responds to the initial GraphiQL editor request with an HTML page that enables testing from the GraphiQL editor.

Disabling the Enable GraphiQL editor option reduces gateway traffic and latency and, furthermore, the Developer Portal provides equivalent test capability regardless of whether or not you enable the option.

This option automatically adds the following property setting to the OpenAPI source for the GraphQL proxy API:

```
html-page: 'store:///graphiql.html'
```

If you disable the Enable GraphiQL editor during API creation and later decide that it is required, you can either manually add this property setting, or you can select Enable GraphiQL editor on the API Setup page of the Design tab of the API..
Note:

- The `html-page` property should be used **only** with a GraphQL proxy API.
- `store:///graphiql.html` is the only value allowed for the `html-page` property.
- If you add the `html-page` property but the Support standard introspection property was not enabled during API creation, the GraphiQL HTML page will not show any documentation regarding the GraphQL backend server, perform query validation, or perform smart lookup. For details on how to manually create the required assembly policy configuration, see Supporting introspection for a GraphQL API.

API Gateway only

Enabling the cost endpoint for a GraphQL API

If you enable the `/graphql/cost` endpoint when creating a GraphQL proxy API, a calling application can obtain details of the cost of executing a request on the backend GraphQL server before making the actual backend request.

The "cost" does not relate purely to monetary cost but can take into account factors such as computational time, memory consumption, and other such factors, in addition to the actual financial costs associated with, say, using third-party systems to retrieve data.

In a production environment, in particular, you should consider carefully the resource implications of making this path available. Note, however, that enabling the `/graphql/cost` endpoint automatically generates policy configuration in your API assembly that would have to be created manually if you disable this option and later decide that it is required; see the API assembly policies for the cost endpoint section for details.

API assembly policies for the cost endpoint

This section details the API assembly policy configuration that is generated automatically if the `/graphql/cost` operation path is selected when a GraphQL proxy API is created. If the path is not selected at creation time, and you later decide that you want to enable it, you must add it manually to the API assembly.

A `case` is added to a `switch` policy with the following condition:

```
($operationPath() = '/graphql/cost' and message.attributes.parse.GraphQLIntrospection = 'not-introspection')
```

If the condition is true, the following policies are executed in sequence, with property settings as indicated:

Rate limit

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

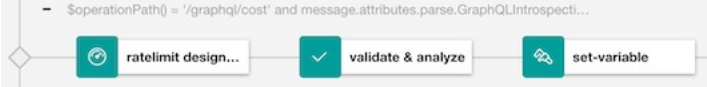
Validate

Property	Value
Input	message
Output	No value, defaults to input
Validate against	GraphQL

Set Variable

Property	Value
Action	Set
Set	message.body
Type	any
Value	$(message.attributes.graphql)

The automatically generated configuration, as it appears in the assembly UI, is as follows:



API Gateway only

Securing a GraphQL API by using a client ID

If you select the Secure using Client ID option when creating a GraphQL proxy API, API Connect automatically adds Rate limit policies to the API assembly.

This section details the Rate limit policy configuration that is generated automatically. If the Secure using Client ID option is not selected at creation time, and you later decide that you want to enable it, you must add the appropriate Rate limit policies manually to the API assembly.

The automatically generated Rate limit policies are added to the **switch** cases in the assembly as follows:

- **switch** case:

  ```
  ($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'not-introspection' and
  message.attributes.graphql.operationType = 'subscription')
  ```

  A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Count Limit Name	graphql-subscription-type-cost
Operation	Increment
Count Limit Name	graphql-subscription-field-cost
Operation	Increment

  The policy as it appears in the assembly UI is as follows:

  This Rate limit policy limits the number of HTML web browser requests for a GraphiQL editor sent to the **/graphql** endpoint.

- **switch** case:

  ```
  ($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'not-introspection' and
  message.attributes.graphql.operationType != 'subscription')
  ```

  Two Rate limit policies with the following property settings:

  - Policy 1:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-type-cost
Operation	Consume
Rate Limit Name	graphql-field-cost
Operation	Consume
Rate Limit Name	graphql-input-type-cost
Operation	Consume

    This Rate limit policy limits calls to the GraphQL proxy API by calculating the expected cost of the query, and then correspondingly reducing the available cost balance that remains of the total configured cost limit. The expected cost is calculating by applying the weighting factors that have been attached to the types and fields in the GraphQL schema; for more information, see Using the GraphQL schema editor.

    The expected cost is calculated by the Validate policy that precedes the Rate limit policy and placed in the API context to be made available to the Rate limit policy. For details on the Validate policy, see Validate.

    The details of the cost calculations to which the referenced rate limits are applied are as follows:

    - **graphql-field-cost**: The weighted sum of fields being selected in a query. Fields that return values of scalar or enum types have a weight of 0.0 and therefore don't affect field cost.
      For example, consider the following query, where it is assumed that the value **max** in the **filter** argument of field **Query.offices** constrains the number of **Office** objects being returned:

      ```
      query {
        offices (filter: {location: "NY", max: 3}) {
          address {
            street
          }
        }
      }
      ```

      The **Address.street** field is of type **String** and therefore has a weight of 0.0. Non scalar and non enum field weights that are not changed in the GraphQL schema editor assume a default value of 1.0. If the **Office.address** field weight has been set to 3.0, the total field cost is 10.0, calculated as follows:

Field	Weight	Field cost
Query.offices	1.0	1 x 1.0 = 1.0
Office.address	2.0	3 x 3.0 = 9.0
Address.street	0.0	0.0
	Total cost:	1.0 + 9.0 + 0.0 = 10.0

    - **graphql-type-cost**:
      The weighted sum of types being returned by a query. Scalar or enum types have a weight of 0.0 and therefore don't affect type cost.

      Considering the previous example, and assuming that all types have a default weight of 1.0, the total type cost is 7 (1 x **Query** type + 3 x **Office** types + 3 x **Address** types.

    - **graphql-input-type-cost**:
      The weighted sum of input types being sent in a query. Scalar or enum input types have a weight of 0.0 and therefore don't affect input type cost.

Considering the previous example, the input type cost is 1.0, because one **`FilterInput`** type is sent in the **`filter`** argument. **`FilterInput.location`** is a **`String`** type and **`FilterInput.max`** is an **`Int`** type, so they are not included as they are scalars.

- Policy 2:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-type-cost
Operation	Replenish
Rate Limit Name	graphql-field-cost
Operation	Replenish

This Rate limit policy limits calls to the GraphQL proxy API by calculating the actual cost of the query, and then correspondingly reducing the available cost balance that remains of the total configured cost limit. The actual cost is calculating by applying the weighting factors that have been attached to the types and fields in the GraphQL schema; for more information, see Using the GraphQL schema editor.

Note that if the actual cost is less than the expected cost, this Rate limit policy will appropriately restore some of the available cost balance that had been removed by Policy 1, thereby overriding the original actual cost limitation. The actual cost might be less than the expected cost if, for example, the query requested more data than was available in a backend database. You can delete one or other of these Rate limit policies from the assembly if desired.
Note: Such unused cost is **not** restored if there are GraphQL response errors. The purpose of this restriction is to prevent malicious requests that cause large amounts of erroneous backend processing while only using a small amount of cost.
The actual cost is calculated by the Validate policy that precedes the Rate limit policy and placed in the API context to be made available to the Rate limit policy. For details on the Validate policy, see Validate.

The policies as they appear in the assembly UI are as follows:

- **`switch`** case:

```
($operationPath() = '/graphql/cost' and message.attributes.parse.GraphQLIntrospection = 'not-introspection')
```

A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

The policy as it appears in the assembly UI is as follows:

This Rate limit policy limits the requests sent to the **`/graphql/cost`** endpoint for retrieving the cost of a GraphQL query.

- **`switch`** case:

```
($operationPath() = '/graphql' and message.attributes.parse.GraphQLIntrospection = 'standard-introspection')
```

A single Rate limit policy with the following property settings:

Property	Value
Source	Plan by Name
Rate Limit Name	graphql-design-request
Operation	Consume

The policy as it appears in the assembly UI is as follows:

This Rate limit policy limits standard introspection requests sent to the **`/graphql`** endpoint.

The rate limits that are referred to by the Rate Limit Name settings in these policies are predefined in Plans that are created in Products that contain one or more GraphQL proxy APIs. You can configure the actual rate limiting values there; for more information, see Editing a draft Product.

For more information on the Rate Limit policy, see Rate Limit.

API Gateway only

Request mechanisms supported by GraphQL endpoints

API Connect supports various mechanisms for sending requests to GraphQL proxy API endpoints.

You can use any of the following mechanisms for sending requests to the **`/graphql`** and **`/graphql/cost`** endpoints on a GraphQL proxy API:

- **GET**: the GraphQL query must be specified in a URL query parameter called **query**.
  For example:

  ```
  https://hostname/basepath/graphql?query={accounts(limit:100){name{first,last}}}
  ```

  The following example includes the **variables** and **operationName** properties, with URL encoding:

  ```
  https://hostname/basepath/graphql?
  query=query%20fetchAccounts%20($limit:%20Int)%20{accounts(limit:%20$limit)%20{name{first,last}}}&variables=
  {"limit":100}&operationName=fetchAccounts
  ```

- **POST** with a JSON-encoded body: the **Content-Type** header must be set to **application/json**, and the GraphQL query must be sent in the **query** key of the JSON object.
  For example:

  ```
  {
    "query": "query fetchAccounts ($limit: Int) {accounts(limit: $limit) {name{first,last}}}",
    "variables": {"limit": 100},
    "operationName": "fetchAccounts"
  }
  ```

  The **variables** and **operationName** are optional. For more information, see POST request in the GraphQL documentation at https://graphql.org/..

- **POST** with graphql content: the **Content-Type** header must be set to **application/graphql**; the entire body of the **POST** is treated as the GraphQL query

Note: For a request sent to the **/graphql** endpoint, the GraphQL proxy API always sends the request to the backend GraphQL server as a **POST** with a JSON-encoded body, so the backend GraphQL server must support this mechanism otherwise the call fails.

GraphQL custom schema directives and scalar types

The API Connect DataPower® API Gateway extends the GraphQL specification by providing additional schema directives and custom scalar types.

Custom schema directives

@cost
: The **@cost** directive associates weight values with types or fields. Types are used to calculate type cost of a request to the GraphQL API. Fields are used to calculate field cost of a request to the GraphQL API. For more information, see The Cost Directive.

@listSize
: The **@listSize** directive applies slicing arguments to fields. For more information, see The List Size directive.

@remove
: The **@remove** GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context. For more information, see @remove directive

@scalarParam
: The **@scalarParam** directive allows further customization of custom scalar types. For more information, see @scalarParam directive.

Custom scalar types

Date
: The **Date** custom scalar type is used to define custom scalars that specify dates. For more information, see Date Custom Scalar.

DateTime
: The **DateTime** custom scalar type is used to define custom scalars that specify date and time. For more information, see DateTime Custom Scalar.

JSON
: The **JSON** custom scalar type is used to specify JSON payloads. For more information, see JSON Custom Scalar.

Long
: The **Long** custom scalar type enables the use of integers larger than those allowed by **Int**. For more information, see Long Custom Scalar.

Related reference

- Using the GraphQL schema editor

@remove directive

The **@remove** GraphQL schema directive specifies conditions for removing types and fields from validation or introspection for each transaction based on values in the API context.

For example, you can prevent specific Plans from using certain types or fields. For more information on the API context, see API Connect context variables.

Note: The **@remove** directive dynamically modifies the schema so that queries that do not conform to the schema, after any **@remove** directives have been evaluated, would be rejected by a **validate** policy.

Syntax

```
@remove(if: [String]) on SCALAR | OBJECT | FIELD_DEFINITION | ARGUMENT_DEFINITION | INTERFACE | UNION | ENUM | ENUM_VALUE | IN
```

Arguments

Argument name	Type	Description
if	List of strings	JSONata condition that specifies the criteria for removal of a type or field from validation or introspection.

Limitations and propagation rules

For limitations and propagation rules, see Limitations and propagation rules for the GraphQL `@remove` directive.

Example 1

Remove the `creditCard` type if the current API plan is `bronze`.

```
type creditCard @remove(if: ["plan.name in [\"bronze\"]"])
```

Example 2

Remove the `creditCard` type if the current API plan is not `gold`, and remove the field `card1` if the API plan is `bronze`.

```
type creditCard @remove(if: ["$not(plan.name = \"gold\")"]){
   field1: String
}
type User {
  card1: creditCard @remove(if: ["plan.name = 'bronze'", "$not(plan.name = \gold\")"])
}
```

API Gateway only

GraphQL limitations

This page details the limitations in the API Connect GraphQL implementation for the current release.

- Specialized handling of custom scalars is not supported, except for validation of the `Long` custom scalar type. Input values provided for all other custom scalar types are passed through.
- The following actions and functions cannot read or write to parsed GraphQL queries. Some of these policies might read and write a serialized form of the GraphQL query.
  - Assembly GatewayScript policy
  - Assembly Map policy
  - Assembly Redaction policy
  - Assembly Set Variable policy
  - Assembly XSLT policy
  - `setVar()` function
- When reading parsed GraphQL queries, the Assembly Switch policy cannot use wildcards or read arguments.
- GraphQL schemas cannot be serialized.
- GraphQL messages received with `Content-Type: application/graphql` or `Content-Type: application/json` will send that same content type to the backend. This behavior cannot be overridden.
- All GraphQL schemas are checked for unbounded lists, including those that are not referenced in the assembly. This behavior can cause validation errors that prevent the GraphQL API from being published.
- The GraphQL schema editor generates warnings for fields that contain unbound lists of values if either of the following conditions is true.
  - The values in the list are of composite type. However, the warning is not valid if the composite type in the list has a defined type weight of 0.0 and all its sub-fields also have weights of 0.0 (both the fields and the types of the values they return).
  - The values in the list are of scalar type, for which a weight other than 0.0 was defined.
- Extending schemas is not supported.
- GraphQL APIs can be tested from the Developer Portal user interface. However, not all authentication methods are supported. You must use one of the following authentication methods:
  - Unsecured
  - Client ID, passed in the header
  - Client ID and secret, passed in the header

  Also, `cost` metrics are not supported, and rate limits specific to GraphQL are not listed in Product plans.

Importing an API from the CP4I asset repository

If you are using IBM® API Connect with IBM Cloud Pak for Integration (CP4I), you can import an API definition from the CP4I asset repository.

About this task

To complete this task you must be assigned a role that has the `Api-Drafts:Edit` permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Procedure

1. In the navigation pane, click Develop, then click Add > API.
   The Select API type screen is displayed.
2. Select OpenAPI 2.0 or OpenAPI 3.0 according to which version of the OpenAPI specification your API is to be based on.
3. In the Import section, select From asset repository, then click Next.
4. Click Launch the asset repository.
5. From the asset repository listing, click the name of the API that you want to import.
   An API explorer window opens, enabling you to examine the details of the API.
6. Click Create from asset.
   The validity of the API definition is checked, and a message is displayed indicating successful validation.
7. Click Next.
8. Optional: Select Activate API if you want to immediately use the API for further development and testing.
   Note:
   - When you select the Activate API option, API Connect automatically completes the following actions:

- Creates a draft product, adds the API to the product, and publishes the product to the Sandbox catalog so that the API is available to be called. The product has the title *api_title* auto product. This product is not visible in the Develop view and you cannot delete it directly. However, if you delete the API the draft product is deleted together with the API; see Deleting an API definition. The product is visible in any catalogs to which it is published. If you want to remove the product from a catalog, you must do this separately; see Removing a product from a catalog
- Subscribes the Sandbox test application to the product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.

- You cannot use the Activate API option if lifecycle approval is enabled in the Sandbox catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to use the Activate API option they must be disabled; for information on lifecycle approval settings, see Creating and configuring catalogs.
- To use the Activate API option, you must be assigned a role that has the `Product:Manage` and `Subscription:Manage` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

9. Click Next.
   The Summary panel displays messages as the API definition is created.
10. Select one of the following options:
    - To further configure your API, click Edit API. For details, see Editing an API definition.
    - If you do not want to configure your API further at this time, click the Develop link in the breadcrumb trail to return to the welcome page; you can then move on immediately to another task. For details on how to configure your API later, see Editing an API definition.

Editing an OpenAPI 2.0 API definition

IBM® API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 2.0 specification.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Procedure

You can open an API for editing in either of the following ways:

- During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
- To open an existing API for editing, complete the following steps:
  1. In the navigation pane, click Develop, then select the APIs tab.
  2. Click the title of the API version that you want to work with.

- **Specifying API metadata**
  API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.
- **Specifying the host for an API**
  You specify a host for an API if you want external developers on the portal to access your APIs by using a host address that is not the same as the API endpoints that are defined for the gateway services where your API will run.
- **Specifying the basepath for an API**
  The base path is the initial URL segment of the API, and does not include the host name or any additional segments for paths or operations. It is shared by all operations in the API.
- **Specifying the schemes for an API**
  Schemes define which transfer protocols you want your API to use. If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.
- **Specifying the MIME types that an API consumes**
  You can specify which types of media your API will accept when calls are made to it. The API Connect gateway supports XML and JSON.
- **Specifying the MIME types that an API produces**
  You can specify which types of media your API will return when calls are made to it. The API Connect gateway supports XML and JSON.
- **Enforcing security requirements on an API**
  To enforce security requirements on an API, you apply previously created security schemes that specify various aspects of API security configuration.
- **Specifying external documentation for an API**
  You can reference an external resource for extended documentation for an API.
- **Adding tags to an API**
  You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- **Defining Paths for an API**
  A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.
- **Defining schema definitions for an API**
  Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.
- **Defining parameters for an API**
  Parameters can specify variable elements of a URL path, query parameters, headers, or a request body.
- **Defining responses for an API**
  Responses define the HTTP status code and data returned in a response body and headers. Responses defined for an API can be used across operations.
- **Specifying gateway and portal settings**
  Define general configuration settings for your API.
- **Defining target services for an API**
  A target service defines a web service that you want to use in your API definition.
- **Setting API properties**
  In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

- **Configuring application authentication for an API**
  Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.
- **Configuring activity logging**
  You can configure your logging preferences for the API activity that is stored in analytics, overriding the default activity logging behavior.
- **Configuring v5 compatibility options**
  You can configure settings to ensure compatibility with API Connect Version 5.0 compatible APIs that are converted for use with the DataPower API Gateway.
- **Including elements in your API assembly**
  An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.
- **Specifying the gateway type for an API definition**
  An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). A default gateway type is set when you create an API definition, but you can edit the API configuration to specify a different gateway type.
- **Converting an API definition for deployment to the DataPower API Gateway**
  IBM API Connect provides two gateway types, DataPower Gateway (v5 compatible) and DataPower API Gateway. If you have an API definition that was developed for the DataPower Gateway (v5 compatible) and you want to port it to the DataPower API Gateway, you must make changes to convert it before deployment.
- **Modifying a GraphQL schema**
  For a GraphQL API, an editor is provided for modifying the associated GraphQL schema.
- **Validating the OpenAPI YAML source**
  How to validate the OpenAPI YAML source against swagger specifications when you're editing an API.
- **Validating an API document by using API governance**
  How to use API governance rulesets to validate and enforce organizational governance policies and best practices. You can validate APIs from the API Manager UI, the API Designer UI, or the toolkit CLI.

Specifying API metadata

API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section, then click Info.
3. Provide the following information:
   - Title (required): The title of the API.
   - Name: The name was generated automatically when the API was created and cannot be changed. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - Version: The API version is specified when the API is initially created and cannot be changed thereafter. You can, however, create a new version of an existing API; see Creating a new version of an API definition.
   - Description: An optional description of the API. You can use CommonMark syntax for rich text representation.
   - Terms of Service: A URL to the Terms of Service for the API.
   - Contact Name: The identifying name of the contact person/organization.
   - Contact Url: A URL pointing to the contact information.
   - Contact Email: The email address of the contact person/organization.
   - License Name: The license name used for the API.
   - License Url: A URL to the license used for the API.
4. Click Save when done.

Specifying the host for an API

You specify a host for an API if you want external developers on the portal to access your APIs by using a host address that is not the same as the API endpoints that are defined for the gateway services where your API will run.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section, then click Host.

3. Enter the required host name.
   Note: To enable this capability, you must additionally configure API vanity endpoints in the Catalog Settings for the Catalog to which the API is published. For information on configuring vanity endpoints, see Creating and configuring Catalogs
4. Click Save when done.

Specifying the basepath for an API

The base path is the initial URL segment of the API, and does not include the host name or any additional segments for paths or operations. It is shared by all operations in the API.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section, then click Base Path.
3. Enter the required base path, preceded by a forward slash character (/).
4. Click Save when done.

Specifying the schemes for an API

Schemes define which transfer protocols you want your API to use. If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section then, if there are already one or more schemes defined, expand Schemes List.
3. To specify a new scheme, click the add icon ⊕ alongside Schemes List. To modify an existing scheme, click the scheme entry in the navigation pane.
4. Select the required scheme from the list provided, then click Create.
5. Click Save when done.

Specifying the MIME types that an API consumes

You can specify which types of media your API will accept when calls are made to it. The API Connect gateway supports XML and JSON.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section then, if there are already one or more consumes MIME types defined, expand Consumes.
3. To specify a new MIME type, click the add icon ⊕ alongside Consumes. To modify an existing MIME type, click the MIME type entry in the navigation pane.
4. Enter the required value in the Mime Type field; for example, **`application/json`** or **`application/xml`**. Then click Create.
5. Click Save when done.

Specifying the MIME types that an API produces

You can specify which types of media your API will return when calls are made to it. The API Connect gateway supports XML and JSON.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section then, if there are already one or more produces MIME types defined, expand Produces.
3. To specify a new MIME type, click the add icon ⊕ alongside Produces. To modify an existing MIME type, click the MIME type entry in the navigation pane.
4. Enter the required value in the Mime Type field; for example `application/json` or `application/xml`. Then click Create.
5. Click Save when done.

Enforcing security requirements on an API

To enforce security requirements on an API, you apply previously created security schemes that specify various aspects of API security configuration.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security scheme definitions, see Defining security schemes.

The following restrictions exist when you apply security schemes to an API:

- You cannot apply more than two API key security schemes to an API.
- If you apply an API key security scheme for client secret, you must also apply an API key security scheme for client ID.
- If you require the application developer to supply both client ID and client secret, you must apply two separate API key security schemes.
- You can have at most one API key scheme of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
- You can have at most one API key scheme of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.
- You cannot apply more than one basic security scheme to an API. If you apply a basic security scheme, you cannot also apply an OAuth security scheme.
- You can apply at most one OAuth security scheme to an API.

A security requirement specifies one or more security schemes whose conditions must all be satisfied for the API to be called successfully. You can define multiple security requirements; in this case, an application can call your API if it satisfies any of the security requirements you have defined.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand General.
3. To create a new security requirement for the API, complete the following steps:
   a. Click the add icon ⊕ alongside Security in the navigation pane.
   b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined; see Defining security schemes.
      If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme; for more information, see Defining OAuth2 security schemes.

      If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower® API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see Configuring scopes for a native OAuth provider.

      Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the `X-IBM-Client-Id` or `client_id` must be included in the security credentials so that the correct Plan configuration settings can be enforced.
   c. Click Create. The security scheme selections are shown; you can change them again before saving.
   d. Click Submit when done.
4. To modify an existing security requirement, complete the following steps:
   a. Click Security in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
   b. To change the security schemes for a security requirement, click the edit icon alongside the required security requirement, then change your security requirement selections as required.

c. Click Submit when done, then click Save.
d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
e. To disable security for the API, clear the Require one of the following Security Requirements check box, then click Save.

- **Defining security schemes**
  A security scheme specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.
- **Enabling CORS support for an API**
  You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

Defining security schemes

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

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create security definitions of the following types:

Type	Description
Basic authentication	Use a basic authentication security definition to specify a user registry or an authentication URL to be used to authenticate access to the API.
API key	Use an API key security definition to specify what application credentials are required to call an API.
OAuth2	Use an OAuth2 security definition to specify settings for OAuth token based authentication for your API.

- **Defining basic authentication security schemes**
  A basic authentication security scheme is used when an application that calls the API is required to authenticate through a user registry.
- **Defining API key security schemes**
  An API key security scheme is used to specify the credentials that an application must provide to identify itself when calling the API operations.
- **Defining OAuth2 security schemes**
  An OAuth2 security scheme defines the settings for controlling access to the API operations through the OAuth authorization standard.

Defining basic authentication security schemes

A basic authentication security scheme is used when an application that calls the API is required to authenticate through a user registry.

Before you begin

For authenticating access to the operations of APIs that are enforced on the API Connect gateway, the following user registry types are supported:

- Authentication URL
- LDAP

The following user registry types are **not** supported:

- OpenID Connect (OIDC)
- Local User Registry (LUR)
- A custom user registry

Before you can create a basic authentication security definition in an API, the user registry must exist. To create a user registry, you can use either API Manager or Cloud Manager. When you create a registry in API Manager, it is visible only to your provider organization. When you create a registry in Cloud Manager, you can make it visible to multiple provider organizations.

To create a user registry with API Manager, see Working with user registries.

To create a user registry with Cloud Manager, see User registries overview.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you use basic authentication, you require API users to provide a valid user name and password to access selected operations. The application developer must also provide an HTTP authorization header in requests that are sent to operations that require basic authentication.

When you use an authentication URL, the user credentials that are provided in the authorization header are validated by the endpoint specified in the URL. If the user is authenticated, IBM® API Connect expects an authentication URL to return an HTTP `200 OK` response status code. All other HTTP response status codes result in an authentication failure and access is denied.

For more information about using an LDAP user registry for authentication, see LDAP authentication.

For information about using an Authentication URL, see Authentication URL user registry.

To make use of a basic authentication security scheme, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there are already one or more security schemes defined, expand Security Schemes.
3. To create a basic authentication security scheme, click the add icon ⊕ alongside Security Schemes, then select basic for the Security Definition Type. To edit an existing basic authentication security scheme, click the security scheme name in the navigation pane.
4. Enter an identifying name of your choice in the Name field.
5. Select the user registry from the Authenticate using User Registry list.
   The user registries in the list are those that have been enabled in the Sandbox Catalog; for details, see Creating and configuring Catalogs.
6. Optionally, provide a Description for the security scheme. You can use CommonMark syntax for rich text representation.
7. If you are creating a new basic authentication security scheme, click Create.
   The security scheme details are displayed for further editing.
8. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

For more information on LDAP and Authentication URL, see LDAP authentication and Authentication URL user registry.

Defining API key security schemes

An API key security scheme is used to specify the credentials that an application must provide to identify itself when calling the API operations.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can require that, when calling an API operation, an application must provide either a client ID, or a client ID and client secret; you create an API key security scheme to specify a credentials requirement. If you require that an application must provide both a client ID and client secret, you must create two API key security schemes, one for each type of credentials.

To make use of an API key security scheme, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there are already one or more security schemes defined, expand Security Schemes.
3. To create an API key security scheme, click the add icon ⊕ alongside Security Schemes, then select apiKey for the Security Definition Type. To edit an API key existing security scheme, click the security scheme name in the navigation pane.
4. Enter an identifying name of your choice in the Security Scheme Name field.
5. Specify the parameter name in the Name field.
   If your API is not enforced by the API Connect gateway, enter the parameter name required by your gateway.

   If your API is enforced by the IBM® API Connect gateway, the required value depends on the gateway type setting for the API, as follows:
   - DataPower® API Gateway: For a key of type client secret, you must include the string "secret", ignoring case, in the value you specify in the Name field; if the value does not contain the string "secret" then the key is assumed to be of type client ID.
   - DataPower Gateway (v5 compatible): the value must be as specified in the following table, depending on where the client credentials are located, and the type of credentials.

   Table 1. Client ID and Client secret parameter name values for the DataPower Gateway (v5 compatible)

Location of credentials	Type of credentials	Parameter name
Header	Client ID	X-IBM-Client-Id
Header	Client secret	X-IBM-Client-Secret
Query	Client ID	client_id
Query	Client secret	client_secret

   When you first create an API, a default API key security definition, of type client ID, is provided.

   For information about including API key parameters in an API call, see Calling an API.
   Note:
   - You cannot apply more than two API key security definitions to an API.
   - If you apply an API key security definition for client secret, you must also apply an API key security definition for client ID.
   - If you require the application developer to supply both client ID and client secret, you must apply two separate API key security definitions.
   - You can have at most one API key definition of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
   - You can have at most one API key definition of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.

- The client ID and client secret headers that are specified in the request when the API is called are **not** added automatically to the message context. If you need these headers in the message context for subsequent processing, include a **`set-variable`** policy in your API assembly that adds the headers to the message content, taking the values specified in the request; you can do this in either of the following ways:
  - In the Assemble tab, add a Set Variable policy to your API assembly that defines the appropriate actions; for example:

    Table 2.

Action	Set	Type	Value
Set	message.headers.X-IBM-Client-Id	string	$(request.headers.X-IBM-Client-Id)
Set	message.headers.X-IBM-Client-Secret	string	$(request.headers.X-IBM-Client-Secret)

    For more information on configuring a Set Variable policy, see Set Variable.
  - In the Source tab, add a **`set-variable`** policy directly to the assembly section in your OpenAPI source; for example:

```
assembly:
  execute:
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: message.headers.X-IBM-Client-Id
            value: $(request.headers.X-IBM-Client-Id)
            type: string
          - set: message.headers.X-IBM-Client-Secret
            value: $(request.headers.X-IBM-Client-Secret)
            type: string
```

    For more information on defining a **`set-variable`** policy in your OpenAPI source, see set-variable.

  You should position the **`set-variable`** before any policy that needs to access the API key headers; for example, an invoke policy that calls a back-end service that is required to identify the application that made the initial request.

6. Select the Key Type: client_id or client_secret.
   Important: The client_id is not confidential and functions as a "SecurityScheme" only in the meaning of the OpenAPI specification.
   If you use only the client_id (without the client_secret) as your API Key, be aware that the client_id is not considered a confidential value in the API Manager UI; for example, it might be displayed to other authenticated members of the provider organization. The client_id is treated as a public identifier for apps and cannot be kept private.

   If authentication of the identity of the app calling an API is needed, then the API Key should use both a client_id to identify the app, plus a client_secret to authenticate it. The client_secret is designed to be known only to the app and the authorization server, and it should be protected. IBM API Connect protects the client_secret: it is only available when first generated, it can be stored as a one-way hash to prevent retrieval or leaks, and it can be changed and reset without changing the corresponding client_id. When an API Key is not marked as a client_secret, it will not be protected in the API Manager UI.

7. Specify whether the credentials are sent in the request header, or as query parameters, by selecting one of the following Located In options:

   header
   : The credentials are sent in the request header. This is the default setting.

   query
   : The credentials are sent as query parameters. This method is less secure because the client secret could be exposed in a log file.

   The selected option is enforced, and API calls fail if the credentials are included in the wrong location by the caller.

   Note:
   - You must specify the same location for the client ID and client secret, either header or query.
   - If you migrated from API Connect V5, API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. For information about the differences, see Client ID behavior in API key security definitions.
8. Optionally, provide a Description for the security scheme. You can use CommonMark syntax for rich text representation.
9. If you are creating a new API key security scheme, click Create.
   The security scheme details are displayed for further editing.
10. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

- **Client ID behavior in API key security definitions**
  How the Client ID behaves in API key security definitions in IBM API Connect 10 compared to Version 5.

Client ID behavior in API key security definitions

How the Client ID behaves in API key security definitions in IBM® API Connect 10 compared to Version 5.

API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. Refer to the following table to understand the differences in Client ID behavior for different use cases.
Note: Although a client_id in the request body is supported, this data is not used during security requirement processing.

Table 1. Client ID behavior in V5 and V10

Use case	Description of security requirement	V5 behavior	V10 behavior
1	Allows client_id in the query, and the request contains multiple client_ids in the query.	Uses the first client ID in the request.	401 invalid client ID or secret.
2	Allows client_id in the header, and the request contains multiple client_ids in the header.	401 invalid client ID or secret.	401 invalid client ID or secret.
3	Allows client_id in the header, and the request contains multiple client_ids in the query and the header (1 in the query, 1 in the header).	401 client ID in wrong location.	Client ID in query is ignored.

Use case	Description of security requirement	V5 behavior	V10 behavior
4	Allows client_id in the query, and the request contains multiple client_ids in the query and the header.	Client ID in header is ignored.	Client ID in header is ignored.
5	Allows client_id in the query or header, and both the query and header contain multiple client_ids.	Header returns 401 client ID in wrong location. Query uses the first client ID in the request.	401 invalid client ID or secret.
6	Allows client_id in the header, and the request contains a valid client_id in the query, and an invalid client_id in the header.	401 client ID in wrong location.	401 invalid client ID or secret.
7	Allows client_id in the query, and the request contains a valid client_id in the header, and an invalid client_id in the query.	401 invalid client ID or secret.	401 invalid client ID or secret.
8	Allows client_id in the query or header, and both the query and header contain valid client_ids.	200 OK.	403 multiple client IDs.
9	Allows client_id in the query or header, and both the query and header contain invalid client_ids.	401 client ID in wrong location.	401 invalid client ID or secret.
10	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the query is valid.	200 OK.	Uses the valid one 200 OK.
11	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the header is valid.	401 client ID in wrong location.	Uses the valid one 200 OK.
12	Allows client_id in the query or header, and both the query and client_id is missing in the request (no security requirement, but client security is defined in the assembly).	401 invalid client ID or secret.	• When Return V5 Responses compatibility option is off: 401 client ID is missing. • When Return V5 Responses compatibility option is on: 401 invalid client ID or secret. For information about setting compatibility options, see Configuring v5 compatibility options.

Defining OAuth2 security schemes

An OAuth2 security scheme defines the settings for controlling access to the API operations through the OAuth authorization standard.

Before you begin

Before you can create an OAuth2 security scheme, you must:

1. Create an OAuth provider.
   - To use Cloud Manager, see Configuring a native OAuth provider or Configuring a third-party OAuth provider.
   - To use API Manager, see Configuring a native OAuth provider or Configuring a third-party OAuth provider.
2. Add the OAuth provider to a catalog. If you have not created any catalogs, use the Sandbox Catalog. See the OAuth instructions step in Creating and configuring Catalogs.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of an OAuth2 security scheme, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. If there are already one or more security schemes defined, expand Security Schemes.
3. To create an OAuth2 security scheme, click the add icon ⊕ alongside Security Schemes, then select oauth2. To edit an existing OAuth2 security scheme, click the security scheme name in the navigation pane.
4. Enter an identifying name of your choice in the Security Scheme Name field.
5. Provide the following information:
   - Security Scheme Name: An identifying name of your choice.
   - OAuth Provider: Select the OAuth provider to be used.
   - Flow: Select one of the following options:
     - Implicit
     - Resource Owner - Password
     - Application
     - Access Code

     Note: The flow options that are available for selection depend on which options are supported by the selected OAuth provider.
     If you selected Resource owner - JWT as the supported grant type when you created the OAuth provider, you must select Resource owner - Password as the Flow for this security scheme.
   - Description: An optional description of the OAuth2 security scheme. You can use CommonMark syntax for rich text representation.

- The URL endpoints (the applicable endpoints depend on the flow type):
  - Authorization Url
  - Token Url

  Note: The URL endpoint values supplied here are maintained only for informative purposes, no validation or other action is applied to them by API Connect. The OAuth security that is applied depends on the specified OAuth provider.
- Scopes (available if you are editing an existing OAuth2 security scheme):
  - To add a new scope, click Add. To edit an existing scope, click the options icon ⋮ alongside the required scope, then click Edit.
  - Provide the following information:
    - Scope: The scope identifier.
    - Description: An optional description of the scope. You can use CommonMark syntax for rich text representation.
    - If you are creating a new scope, click Create.

6. If you are creating a new OAuth2 security scheme, click Create.
   The security scheme details are displayed for further editing.
7. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

Enabling CORS support for an API

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

Before you begin

This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.

About this task

Note:

- CORS support is available only on the DataPower® API Gateway.
- When CORS is enabled, the API Gateway runs the **`cors`** preflow policy to handle all CORS requests that are made to the API.
- When CORS is enabled and a preflight request is received, only the following API actions are performed:
  - The **`cors`** preflow policy configures the appropriate response headers.
  - The response headers are set.
- When a preflight request is received, the **`request.attributes.isCORSPreflight`** flag is set to **`true`**.
- For all preflight requests, the **`security`** and **`client-identification`** preflow policies are always skipped, whether CORS is enabled or not enabled.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

To enable CORS support for an API, complete the following steps:

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings, then click CORS.
   The CORS page opens.
3. Select Enabled.
4. Optional: Configure a CORS policy.
   Should you create a CORS policy? Review the following considerations:
   - A CORS policy is an optional part of an API definition. If an API definition has no CORS policy but CORS is enabled, then CORS requests will be accepted from all Origins.
     If you want to accept CORS requests from all Origins, then enable CORS but do not add a CORS policy to the API definition.
   - If you create a CORS policy, then CORS requests will only be accepted from Origins that are explicitly listed in the CORS rules contained in the CORS policy. CORS requests from any other Origins will be rejected.
     If you want to accept CORS requests from only a limited number of Origins (and might also want to configure the Access-Control-Allow-Credentials and Access-Control-Expose-Headers response headers), then you should enable CORS and create a CORS policy. Only those Origins explicitly listed in the allow-origin field of the CORS rules in the CORS policy will be accepted; CORS requests from any Origins not listed will be rejected.
   - To configure a new CORS policy, complete the following steps:
     - Alongside Policy, click Add.
     - To include the header **`Access-Control-Allow-Credentials: true`** in a response. select Allow Credentials. The **`Access-Control-Allow-Credentials`** response header tells browsers whether to expose the response to front-end JavaScript code when the request's credentials mode (**`Request.credentials`**) is set to **`include`**.
     - To append one or more of the following values to the **`Access-Control-Expose-Headers`** response header, select Expose headers, and select from the following options:
       - Predefined - The predefined value of the Gateway. This option is selected by default.
       - Backend - The value of **`Access-Control-Expose-Headers`** from the backend response.
       - Custom - A custom string.
     - Click Create.

- Alongside Allowed Origins click Add.
- Enter the origin URL, then click Create. This setting indicates that the response can be shared with requesting code from the specified origin.

- To modify an existing CORS policy, click its entry on the CORS page. You can then modify individual origin URLs, add further origins, and change the Allow Credentials setting.

Example

The following example has three rules:

- Accept an Origin header of `https://example.com`, and return `Access-Control-Allow-Credentials: true` in the CORS response.
- Accept an Origin header of `http://domain.com`. No `Access-Control-Allow-Credentials` header will be returned in the response.
- Accept any of the following Origin headers:
  - `http://example2.com`
  - `http://example3.com`
  - `http://example4.com`

  No `Access-Control-Allow-Credentials` header will be returned in the response.

```
cors:
  enabled: true
  policy:
    -
      allow-origin:
        - 'https://example.com'
      allow-credentials: true
    -
      allow-origin:
        - 'http://domain.com'
    -
      allow-origin:
        - 'http://example2.com'
        - 'http://example3.com'
        - 'http://example4.com'
```

5. Click Save to save your changes.
6. Optional: To implement your own CORS solution using custom OPTIONS operations, complete the following steps:
   a. Add the following headers to your HTTP responses:

   ```
   Access-Control-Allow-Origin: https://<portalhostname>
   Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
   ```

   Where `<portalhostname>` is your Developer Portal host name.
   b. Optional: You can proxy your API through API Connect as an enforced invoke API so that CORS is handled automatically.

Important:

- If you implement your own CORS solution, you **must** disable the CORS option described in step 3
- CORS preflight requests are sent by using the HTTP `OPTIONS` method. Therefore, if you require these requests to be handled by the API Connect gateway then you must enable the `OPTIONS` method for all APIs that will handle preflight requests; see Defining Paths for a REST API.
- `OPTIONS` requests are counted as API calls against any configured rate limit. Note that you can apply rate limits to individual operations; see Defining rate limits for an API operation.

Specifying external documentation for an API

You can reference an external resource for extended documentation for an API.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General section, then click External Documentation.
3. Provide the following information:
   - URL (required): The URL for the target documentation.
   - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
4. Click Save when done.

Adding tags to an API

You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand the General then, if there already one or more tags defined, expand Tags.
3. To add a new tag, click the add icon ⊕ alongside Tags in the navigation pane. To edit an existing tag, click the tag name in the navigation pane.
4. Provide the following information:
   - Name (required): The name of the tag.
   - Description: An optional description of the tag. You can use CommonMark syntax for rich text representation.
   - External Documentation > URL (available when editing an existing tag): The URL for target documentation that describes this tag.
   - External Documentation > Description (available when editing an existing tag): An optional description of the target documentation.
5. If you are creating a new tag, click Create.
6. Click Save when done.

Defining Paths for an API

A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there are already one or more Paths defined, expand Paths.
3. To create a new Path, click the add icon ⊕ alongside Paths in the navigation pane. To edit an existing Path, click the Path in the navigation pane.
4. Provide the following information:
   - Path: A relative path to an individual endpoint.
     The path is appended to the base path to construct the full URI to access the APIs. The path must start, but not end, with the / character. A parameter at the end of the path can contain a qualifier to match one or more path levels.

     If you specify just the name of the parameter, then one level of that path is matched. If you want to allow for multiple levels of the path, you can prefix the parameter with one of the following qualifiers:

     - * to indicate 0 or more occurrences
     - + to indicate 1 or more occurrences

     The + and * qualifiers can only be used at the end of the path.

     For example, the path:

     ```
     /petstore/{type}/{*category}
     ```

     matches the following paths, where only one type value is matched, but all (0 or more) categories are matched:

     ```
     /petstore/cats
     /petstore/cats/supplies
     /petstore/cats/supplies/health
     /petstore/cats/supplies/health/medicines
     /petstore/cats/supplies/health/medicines/a/b/c
     ```

   - $ref (available when editing an existing Path): A reference to a Path that is defined elsewhere, either in this OpenAPI definition, or in an external file. A reference enables reuse of a set of operations already defined for a Path in this API, or in a file that is either a complete API definition or an OpenAPI fragment.
     The reference uses JSON pointer notation. For example, a reference to a Path called `/mypath` in this API would have the following format:

     ```
     #/paths/~1mypath
     ```

     The `~1` characters are used to escape the `/` character, which is being used literally in the path name rather than as a special character.
     A reference to a path in an external file called path_fragments.yaml would have the following format:

     ```
     file://path_fragments.yaml/paths/~1mypath
     ```

5. If you are creating a new Path, click Create.
   The Path details are displayed for further editing as described in step 4.
6. Click Save when done.

- **Defining parameters for a path**
  Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

- **Defining operations for a path**
  Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

Defining parameters for a path

Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths.
3. Expand the required Path, then, if there already one or more parameters defined, expand Path Parameters.
4. You can either create a new parameter, or edit any existing parameter.
   - To create a parameter, click the add icon ⊕ alongside Path Parameters, then refer to Creating a parameter.
   - To edit an existing parameter, click the parameter name in the navigation pane, then refer to Editing a parameter.

Defining operations for a path

Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required path.
3. Click the add icon ⊕ alongside Operations under the Path in the navigation pane.
4. Provide the following information:
   - Verb: Select the operation type from the following options:
     - get
     - post
     - put
     - delete
     - options
     - head
     - patch
     - trace
   - Summary: A summary of what the operation does.
   - Description: A description of the operation behavior.
   - Operation: Unique string used to identify the operation. The id must be unique among all operations described in the API.
   - Deprecated: Declares this operation to be deprecated. Consumers should refrain from using the declared operation.
5. Click Create.
6. Click Save when done.

- **Adding tags to an operation**
  You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- **Specifying external documentation for an operation**
  You can referencing an external resource for extended documentation for an API operation.
- **Specifying the MIME types that an API operation produces**
  You can specify which types of media your API operation will return when calls are made to it. The API Connect gateway supports XML and JSON.
- **Specifying the MIME types that an API operation consumes**
  You can specify which types of media your API operation will accept when calls are made to it. The API Connect gateway supports XML and JSON.
- **Defining parameters for an operation**
  Responses define the HTTP status code and data returned in a response body and headers.

- **Defining responses for an operation**
  Responses define the HTTP status code and data returned in a response body and headers. A response defined for an operation is one that can returned by executing the operation.
- **Specifying schemes for an operation**
  The schemes settings define which transfer protocols you want your API operation to use.
- **Enforcing security requirements on an operation**
  To enforce security requirements on an API operation, you apply previously created security schemes that specify various aspects of API security configuration.

Adding tags to an operation

You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To add a new operation tag, click the add icon ⊕ alongside the Tags entry for the operation in the navigation pane. To edit an existing tag, expand Tags then click the tag name in the navigation pane.
5. Provide the following information:
   - Name (required): The name of the tag.
6. If you are creating a new operation tag, click Create.
7. Click Save when done.

Specifying external documentation for an operation

You can referencing an external resource for extended documentation for an API operation.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click External Documentation.
4. Provide the following information:
   - URL (required): The URL for the target documentation.
   - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
5. Click Save when done.

Specifying the MIME types that an API operation produces

You can specify which types of media your API operation will return when calls are made to it. The API Connect gateway supports XML and JSON.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To specify a new MIME type, click the add icon ⊕ alongside Produces. To modify an existing MIME type, click the MIME type entry in the navigation pane.
5. Enter the required value in the Mime Type field; for example **`application/json`** or **`application/xml`**. Then click Create.
6. Click Save when done.

Specifying the MIME types that an API operation consumes

You can specify which types of media your API operation will accept when calls are made to it. The API Connect gateway supports XML and JSON.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To specify a new MIME type, click the add icon ⊕ alongside Consumes. To modify an existing MIME type, click the MIME type entry in the navigation pane.
5. Enter the required value in the Mime Type field; for example, **`application/json`** or **`application/xml`**. Then click Create.
6. Click Save when done.

Defining parameters for an operation

Responses define the HTTP status code and data returned in a response body and headers.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. You can either create a new operation parameter, or edit any existing parameter.
   - To create an operation parameter, click the add icon ⊕ alongside Path Parameters, then refer to Creating a parameter.
   - To edit an existing operation parameter, click the parameter name in the navigation pane, then refer to Editing a parameter.

Defining responses for an operation

Responses define the HTTP status code and data returned in a response body and headers. A response defined for an operation is one that can returned by executing the operation.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing responses, expand Responses.

4. You can either create a new response, or edit any existing response.
   - To create a response, click the add icon ⊕ alongside Responses, then refer to Creating a response.
   - To edit an existing response, click the response name in the navigation pane, then refer to Editing a response.

Specifying schemes for an operation

The schemes settings define which transfer protocols you want your API operation to use.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing schemes, expand Schemes List.
4. You can either add a new scheme, or edit any existing scheme.
   - To add a scheme, click the add icon ⊕ alongside Schemes List, select the required transfer protocol, then click Create.
     Note: If your API is enforced by an API Connect gateway, only the HTTPS protocol is supported.
   - To edit an existing scheme, click the scheme in the navigation pane, then change the transfer protocol as required.
5. Click Save when done.

Enforcing security requirements on an operation

To enforce security requirements on an API operation, you apply previously created security schemes that specify various aspects of API security configuration.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security schemes, see Defining security schemes.

A security requirement specifies one or more security scheme definitions whose conditions must all be satisfied for the API operation to be called successfully. You can define multiple security requirements; in this case, an application can call your API operation if it satisfies any of the security requirements you have defined.

Any security requirements that you define for an operation completely override any security requirements defined on the parent API. If you do not define any security requirements for an operation, or you delete all security requirements from an operation, the operation inherits the security requirements defined on the parent API. For more information, see Enforcing security requirements on an API.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To create a new security requirement for the operation, complete the following steps:
   a. Click the add icon ⊕ alongside the Security entry for the operation in the navigation pane.
   b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have previously been defined; see Defining security schemes.
      If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme; for more information, see Defining OAuth2 security schemes.

      If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower® API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see Configuring scopes for a native OAuth provider.

      Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the **`X-IBM-Client-Id`** or **`client_id`** must be included in the security credentials so that the correct Plan configuration settings can be enforced.
   c. Click Create. The security scheme selections are shown; you can change them again before saving.
   d. Click Save when done.
5. To modify an existing security requirement, complete the following steps:
   a. Click the Security entry for the operation in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.

b. To change the security schemes for a security requirement, click the edit icon alongside the required security requirement, then change your security requirement selections as required.
c. Click Submit when done, then click Save.
d. To delete a security requirement, click the appropriate delete icon, click Delete to confirm, then click Save.
e. To disable security for the operation, clear the Require one of the following Security Requirements check box, then click Save.

Note: These settings completely override any security requirements defined on the parent API; see Enforcing security requirements on an API.

Defining schema definitions for an API

Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
An API created in OpenAPI 2.0 includes a section where API payload definitions are defined. The payload definitions describe the structure of data transmitted in API requests and responses, utilizing a specialized JSON schema variant specific to OpenAPI. Each payload definition corresponds to a schema, which lists the expected data fields. These data fields are represented as properties within the schema and include a set of attributes that describe their type, permissible values, mandatory status, and other characteristics. Although numerous attributes are available, the majority of them are infrequently utilized.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there are already one or more schemas defined, expand Definitions.
3. You can create a new schema, or edit any existing schema.
   - To create a schema, click the add icon alongside Definitions, then refer to Creating schema definitions.
   - To edit an existing schema, click the schema component name in the navigation pane, then refer to Editing schema definitions.

- **Creating schema definitions**
  Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.
- **Editing schema definitions**
  Schemas provide developers with information about the request that they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

Creating schema definitions

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.

Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.
For details of the areas in your API definition from where you can create a schema, see the following topics:

- Defining schema definitions for an API
- Editing a response

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon.

Procedure

1. To create a schema definition, click the add icon alongside Definitions.
   The Add object wizard for a schema is displayed.
2. Provide the following information:
   - Schema Name: This name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/definitions/`*`Name`*

- Title: The schema title.
- Type: The schema data type; select one of the following:
  - array
  - boolean
  - integer
  - number
  - object
  - string

3. Click Create.
   The schema definitions details are displayed for further editing; see Editing schema definitions.

Editing schema definitions

Schemas provide developers with information about the request that they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.
For details of the areas in your API definition from where you can edit a schema, see the following topics:

- Defining schema definitions for an API
- Editing a response

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The table view presents the properties within a schema alongside their essential attributes, such as name, type, description, and required status. The secondary pop-up enables access for viewing and editing the more detailed and less commonly used attributes that are associated with each data field.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

You can configure the following schema definition settings:

- Schema Name: This name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

  `#/definitions/Name`

  To change the name, click the Source icon, change the value in the field, then click Save.
- Optional: Title: The schema title.
- Type: The schema data type; select one of the following:
  -
    - array
    - boolean
    - integer
    - number
    - object
    - string

  Note: If you change the type of an existing schema, any existing settings that are not relevant to the new type are still retained in the OpenAPI source even though they are no longer displayed in the user interface. You should therefore avoid porting a schema from one type to another but instead either create a new schema or modify the OpenAPI source directly; to open the source editor, click the Source icon </> .
- Properties: The schema property. To add a property to the schema, click Add alongside the Properties section heading, complete the details and click Save.
- Additional schema settings dependent on the selected Type of property. To view and edit these schema settings, click the Options menu next to the property you want to view and select Details.
  - boolean: no type specific settings.
  - integer:
    - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
    - Multiple Of: Division of an integer instance by this value must result in an integer.
    - Maximum: An integer instance must be less than or equal to this value.
    - Exclusive Maximum: An integer instance must be strictly less than this value.
    - Minimum: An integer instance must be greater than or equal to this value.
    - Exclusive Minimum: An integer instance must be strictly greater than this value.
  - number:
    - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
    - Multiple Of: Division of a number instance by this value must result in an integer.
    - Maximum: A number instance must be less than or equal to this value.
    - Exclusive Maximum: A number instance must be strictly less than this value.
    - Minimum: A number instance must be greater than or equal to this value.
    - Exclusive Minimum: A number instance must be strictly greater than this value.

    - object:
        - Max Properties: The number of properties in an object instance must be less than or equal to this value.
        - Min Properties: The number of properties in an object instance must be greater than or equal to this value.
        - Required Properties: An array of property names; every property name in the array must be the name of a property in an object instance. To add a required property name, click Add alongside the Required Properties section heading, enter the name, then click Create.
    - string:
        - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
        - Pattern: A regular expression; a string instance must successfully match the expression.
        - Max Length: The maximum allowed number of characters in a string instance.
        - Min Length: The minimum allowed number of characters in a string instance.
        - Enum: An array of string values; a string instance must match one of the array values. To add an enum value, click Add, enter the value, then click Create.
- General settings; select the required options:
    - Required: A property associated with this schema is required.
    - Read Only: A property associated with this schema can be returned in a response but should not be included in a request. For example, properties whose values are system generated identifiers will be defined as Read Only because application clients wouldn't be able to set these.
- Example Value: An example value. Whatever you enter here is displayed as-is in the Developer Portal.
- External Documentation: An external resource for extended documentation; provide the following information:
    - URL (required): The URL for the target documentation.
    - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
- Xml: Metadata that allows for more fine-tuned XML model definitions; provide the following information:
    - Name: Replaces the name of the element or attribute that is used for the described schema property.
    - Namespace: The URL of the namespace definition.
    - Prefix: The prefix to be used for the Name.
    - Attribute: Specifies whether the property definition translates to an attribute instead of an element.
    - Wrapped: For an array definition, Specifies whether the array is wrapped (for example, `<books><book/><book/></books>`) or unwrapped (`<book/> <book/>`).
- Click Save when done.

Example

By defining properties, you can create more complex data structures. For example, you could define an "Address" schema that has the following properties:

PROPERTY NAME	PROPERTY TYPE
street1	string
street2	string
city	string
state	string
zip_code	string

and then define a "BankBranch" schema that has the following properties, one of which is a reference to the Address schema:

PROPERTY NAME	PROPERTY TYPE
address	address
type	string
id	string

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining parameters for an API

Parameters can specify variable elements of a URL path, query parameters, headers, or a request body.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there already one or more parameters defined, expand Parameter Definitions.
3. You can either create a new parameter, or edit any existing parameter.
    - To create a parameter, click the add icon alongside Parameters, then refer to Creating a parameter.
    - To edit an existing parameter, click the parameter name in the navigation pane, then refer to Editing a parameter.

- **Creating a parameter**
  Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.
- **Editing a parameter**
  Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for the API, or for Paths and Path operations in your API definition.

Creating a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.

Before you begin

Launch the parameter creation window. For details of the areas in your API definition from where you can create a parameter, see the following topics:

- Defining parameters for a path
- Defining parameters for an operation
- Defining parameters for an API

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Parameter Name: The name of the parameter.
   - Located In: The location of the parameter. Select one of the following options:
     - query: Parameters that are appended to the URL. For example, ***`url_path`*`?myparam=myvalue`**.
     - header: Custom headers that are expected as part of the request.
     - path: The parameter value is part of the operation's URL, enclosed in {}.
     - body: Parameters that describe the body of POST, PUT and PATCH requests.
   - Description: A description of the parameter.
   - Type: The data type of the parameter.
   - Format: Select the data format; the available options depend on the selected Type.
   - Select the following option as required:
     - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
     - Allow Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
2. Click Create.
   The parameter details are displayed for further editing; see Editing a parameter.

Editing a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for the API, or for Paths and Path operations in your API definition.

Before you begin

Open the details form for a parameter. For details of the areas in your API definition from where you can edit a parameter, see the following topics:

- Defining parameters for a path
- Defining parameters for an operation
- Defining parameters for an API

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Description: A description of the parameter. You can use CommonMark syntax for rich text representation.
   - Type: The data type of the parameter.
   - Format: Select the data format; the available options depend on the selected Type.
   - Select the following options as required:
     - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
     - Allow Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining responses for an API

Responses define the HTTP status code and data returned in a response body and headers. Responses defined for an API can be used across operations.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. If there already one or more responses defined, expand Response Definitions.
3. You can either create a new response, or edit any existing response.
   - To create a response, click the add icon ⊕ alongside Response Definitions, then refer to Creating a response.
   - To edit an existing response, click the response name in the navigation pane, then refer to Editing a response.

- **Creating a response**
  Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.
- **Editing a response**
  Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

Creating a response

Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.

Before you begin

Launch the response creation window. For details of the areas in your API definition from where you can create a response, see the following topics:

- Defining responses for an API
- Defining responses for an operation

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: The HTTP status code.
   - Description (required): A description of the response.
2. Click Create.
   The response details are displayed for further editing; see Editing a response.

Editing a response

Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

Before you begin

Open the details form for a response. For details of the areas in your API definition from where you can edit a response, see the following topics:

- Defining responses for an API
- Defining responses for an operation

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: The HTTP status code.
   - Description (required): A description of the response.
   - Schema: The schema defines the response structure.
     To define a schema for the response, click Create, then refer to Creating schema definitions.

     If the response already has a schema defined, click View to edit the schema.

     For full details on editing a schema, see Editing schema definitions.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Specifying gateway and portal settings

Define general configuration settings for your API.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, then click Gateway and portal settings.
3. Provide the following information:
   - Target Services: Define web services that you want to use in your API definition. For details, see Defining target services for an API.
   - Enforced: Select this option to enforce the API by using the DataPower® API Gateway. Clear this option if you are managing the API on a gateway other than the DataPower API Gateway. Although not published to a gateway by API Connect, an unenforced API is still available in the Developer Portal for subscription by application developers.
   - Phase: The phase of the lifecycle that your API is in; select one of the following options:
     - Realized (default) - The API is in the implementation phase.
     - Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
     - Specified - The API has been fully designed and passed an internal milestone but has not yet been implemented.
   - Testable: Select this option to allow the API's operations to be tested using the test tool in the Developer Portal.
     Note: For the test tool to work, an API must be included in a Plan in a Product that is staged in a Catalog.
   - CORS: Enable and configure cross-origin resource sharing (CORS) support for your API. For details, see Enabling CORS support for an API.
   - Properties: Add properties that can be referenced in your API definition. For details, see Setting API properties.
   - Catalog Properties: Define property values that are specific to a particular Catalog. For details, see Defining Catalog specific property values.
   - Gateway: For details, see Specifying the gateway type for an API definition.
   - Application Authentication: Enable application authentication to protect your API with a certificate. For details, see Configuring application authentication for an API.

- Application Authentication Source: If application authentication is enabled, specify how the client certificate is sent to the gateway. For details, see Configuring application authentication for an API.
- Activity Log: Configure your logging preferences for the API activity that is stored in analytics. For details, see Configuring activity logging.
- Compatibility: For details, see Configuring v5 compatibility options.
- Html Page: For a GraphQL API, the HTML page with which the API responds to an initial GraphiQL editor request. Currently, the only option is `store:///graphiql.html`. For more information, see Enabling the GraphiQL editor for a GraphQL API.
- Graphql Schema: For a GraphQL API, the name of the schema definition that references the imported GraphQL schema. For information on uploading a schema during GraphQL API creation, see Creating a GraphQL proxy API. For information on replacing the schema for an existing GraphQL API, see Using the GraphQL schema editor.
- Exxtensions: Extensions added to the API to extend the OpenAPI specification. For details, see Adding an OpenAPI extension to an API.

Defining target services for an API

A target service defines a web service that you want to use in your API definition.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more target services defined, expand Target Services.
3. To define a new API target service, complete the following steps:
   a. Click the add icon alongside Target Services in the navigation pane.
   b. Upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, by either dragging and dropping your file, or browsing and selecting the file that you want to use.
   If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see Using an options file when importing a WSDL service.

   The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see Including elements in your API assembly.
4. Select the service that you want to add.
5. Click Save to save your changes.
6. To view the source for an existing target service, click the target service name in the navigation pane.

Results

The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see Including elements in your API assembly.

Setting API properties

In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API properties include property name, value, and, optionally a specific Catalog to which a property value applies. For a list of pre-supplied API properties relating to various policies, see API properties.
Note: Once defined, an API property is read only.
For information on how to reference a property in an API definition, see Variable references in API Connect.

It is also possible to define properties that are specific to a Catalog and can be referenced by any of the APIs in that Catalog; for more information, see Creating and configuring Catalogs. Note that if you define a Catalog property of the same name as an API property, the API property takes precedence over the Catalog property.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more API properties defined, expand Properties.
3. Configure an API property.
   - To define a new API property, complete the following steps:
     - Click the add icon alongside Properties in the navigation pane.
     - Provide the following information:
       - Property Name: Enter a name for the property; this name is used to reference the property. The following character set is supported for the name of an API property: `[A-Za-z0-9_-]+`. Spaces are allowed.
       - Property Value: A default value for the property. Leave blank if the property is to have a null vale by default.
       - Description: An optional description of the property. You can use CommonMark syntax for rich text representation.
       - Select Property Value is Base64 Encoded if you want to hide the property values, or protect user passwords from casual observance.
         Note: If you encode a property value, it is saved in Base64 encoded form; it is **not** encrypted. If you subsequently clear the Encoded check box, the original property value is restored in its unencoded form.
     - Click Create.
   - To modify an existing API property, click the property name in the navigation pane. You can then change any of the configuration settings.
4. Click Save to save your changes.

- **Defining Catalog specific property values**
  For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

Defining Catalog specific property values

For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

Before you begin

You must have previously created the API property that you now want to define a Catalog specific value for; see Setting API properties.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more Catalog specific property values defined, expand Catalog Properties.
3. Configure a Catalog specific property value.
   - To define a new Catalog specific property value, complete the following steps:
     - Click the add icon alongside Catalog Properties in the navigation pane.
     - Type the Catalog Name exactly as it displays in the **`name`** field for the selected Catalog.
       Tip: For the best result, manually type the catalog's name instead of selecting its title. Using the catalog name is safer than using its title, because the title might not be unique within the provider organization and might be modified later (which breaks the association between the property and the catalog). The catalog's name is always unique within the provider organization and cannot be changed.
     - Click Create. The Catalog properties details for the specified catalog are displayed for further editing.
     - Alongside Property Overrides, click Add.
     - Provide the following information:
       - Property Name to Override: Enter the name of the API property that you want to define a Catalog specific value for. The name **must** match the name of a previously defined API property; see Setting API properties.
       - Property Value: The Catalog specific property value. Leave blank if the property is to have a null value by default.
     - Click Create.
   - To modify an existing Catalog specific property value, click the Catalog name in the navigation pane. You can then change the Catalog name, and add, modify, or delete property override settings.
4. Click Save to save your changes.

Configuring application authentication for an API

Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Enable application authentication, as follows:
   Note: To identify the application, you must enable a security requirement of type **`clientID`**.
   a. Select the Gateway tab, expand Gateway and portal settings, then click Application Authentication.
   b. Select Certificate.
3. Specify how the client certificate is sent to the gateway, as follows:
   a. In the navigation pane, click Application Authentication Source.
      Any existing application authentication source definitions are listed.
   b. Alongside Application Authentication Source, click Add.
   c. Select the required option, header or tls-cert.
      You can separately add one or both of the options.
      When the API is called, an X509 client certificate must be supplied, either in the specified HTTP header, or as a TLS client certificate from TLS mutual authentication. For any Developer Portal application that calls the API, the certificate must be entered in the Developer Portal user interface; for details, see Registering an application.

      The Gateway service to which the API is published can be configured to use TLS mutual authentication to secure API calls made to that Gateway service; for details, see Configuring the initial Gateway service. If you select this option, continue to sub-step d.

      If you are using a load balancer, you must configure the load balancer to use the specified HTTP header to relay the appropriate client certificate to the Gateway service after the load balancer terminates the TLS communication.

   d. If you select the tls-cert option, enable mTLS at the Gateway level.
      This can be done by enabling "Mutual authentication: Required" in the TLS Server Profile configured in the Gateway Service in the Cloud Manager Topology (requires cloud administrator permissions). For information, see Registering a gateway service.

      You can create a Truststore (with the root or intermediate CA certificate) for use with the TLS Server Profile.

      The following example cURL command uses a client-cert during SSL:

      ```
      **curl -X GET 'https://GW-Server:9443/test/sandbox/mtls-testing/callmtls-call' -H 'X-IBM-Client-Id:
      1872xxxxxxxxxxxx' --cert my-leaf-cert.pem --key my-leaf-cert.key -k -v**
      ```

      Note: This feature is intended for the scenario where different applications will be registered with different end certificates in the Developer Portal, which will have the same intermediate or root CA issuer. The certificate will pass Gateway-level mTLS if it matches the issuer available in the Truststore, but will only allow access the to API that matches the correct end certificate.
4. Click Create.
5. Click Save to save your changes.

Configuring activity logging

You can configure your logging preferences for the API activity that is stored in analytics, overriding the default activity logging behavior.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An API event record exists for each API execution event in the Gateway server. By default, the content type for a successful API execution is activity, and payload for an API call the results in an error code. When you compose your API definition, you can change the type of content to log in these API event records. During API execution, the activity data is stored in the **`log`** context variable, which populates the API event record on completion of the API execution; for more information, see API activity logging context variables.

Note:
Activity logging that calls for logging of analytics data upon success doesn't apply for the OAuth provider. The OAuth provider logs analytics data for failure cases, but doesn't log successful cases.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings, then click Activity Log.
3. Select Enabled, then select options for the Success Content and Error Content fields as follows:

Field label	Description

Field label	Description
Success Content	Defines the type of content to be logged when the operation is successful. • none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • activity: Logs invocation only; only the resource URI is recorded. • header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded. • payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded. The default value is activity.
Error Content	Indicates what content to log when an error occurs. • none: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • activity: Logs invocation only; only the resource URI is recorded. • header: Logs activity and header; the HTTP headers, in addition to the resource URI are recorded. • payload: Logs activity, header, and payload; all information, including the payload that is received in a request or returned in a response, is recorded. The default value is payload.
Preserved request header	Click Add to specify customized request headers to preserve for logging to analytics.
Preserved response header	Click Add to specify customized response headers to preserve for logging to analytics.

4. Click Save to save your changes.

Configuring v5 compatibility options

You can configure settings to ensure compatibility with API Connect Version 5.0 compatible APIs that are converted for use with the DataPower API Gateway.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings, then click Compatibility.
3. Select from the following options as required:
   - Return V5 Responses: Return v5-compatible responses, such as OAuth and client security error responses.
   - Copy Id Headers to Message: Copy security headers to the message context for retrieval by the invoke back-end service.
   - Enforce Required Params: Check the request for required parameters during API routing. When enabled, a request that does not provide a required parameter is rejected.
   - Allow Chunked Uploads: Allow the assembly **`invoke`** policy to send documents to the server with **`Transfer-Encoding: chunked`**. This setting applies only to the **`invoke`** 1.5.0 wrapper policy deployed from API Connect using the migration utility. It does not apply to the native API Gateway assembly **`invoke`** policy.
4. Use the YAML source to set the following options as required:

Option	Valid values	Description
`allow-trailing-slash`	**`true / false`**	Specify whether to allow the request URL to end with a / character. The default value is **`false`**.
`get-raw-body-value`	**`true / false`**	Specify whether the GatewayScript **`apim.getvariable()`** API returns the raw body instead of parsing. This setting applies only when the context is other than **`message`**. The default value is **`false`**.
`request-headers`	**`true / false`**	Specify whether to populate v5-compatible headers such as **`X-Client-IP`** and **`X-Global-Transaction-ID`** in the **`request.headers`** context variable. The default value is **`false`**.

5. Click Save to save your changes.

Including elements in your API assembly

An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
In the Policies view of IBM® API Connect, you can add and configure elements in your assembly. You can also directly add elements to the OpenAPI definition of your API.

- **The assembly editor**
  The API Designer of IBM API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

- **Adding elements to your assembly**
  Create your API assembly by using the Assemble view in the API Designer. You can access the Assemble view by clicking the Gateway tab in the API editor.
- **Handling errors in the assembly**
  Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

The assembly editor

The API Designer of IBM® API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can access the Assemble view by clicking the Gateway tab in the API editor, and then click Policies. This tab includes the following options:

- Palette - lists the available elements in the assembly flow.
- Canvas - used to arrange and visualize the elements in the assembly.
- Property sheet - used to configure the elements in the assembly.

The palette

The palette is a list of different elements that you can include in your assembly. To view the palette, click the plus icon ⊕ on the canvas. You can use the Rotate icon ⟳ to rotate the palette as you prefer. To maximize and minimize the palette view, click the Maximize and Minimize icons.

You can release the palette by clicking the Close icon ✕.

The canvas

You can use the canvas to create a graphical representation of the assembly flow. To add an element to the assembly flow from the palette, click the plus icon ⊕ at the appropriate location on the canvas and select the required element from the available list. To delete an element from the canvas, hover over the element and click the Delete icon 🗑.

Note: You must click Save in the Assemble view to save your assembly updates, and to make them appear in the Source view.
API calls are made at the initial, unfilled, circle, and returned at the final, filled, circle. You can insert elements between the two circles to modify the data received from the call, or returned by the response.

You can use the Show catches toggle to show and hide error catches in the palette. A catch is a section of the assembly that is applied when an API call results in the corresponding HTTP status code being returned. Click the catch section to open the property sheet for all your catches.

You can zoom the view of your canvas in and out by clicking the + and - icons. To fit the canvas to the screen size, click the Fit to screen icon ⛶.

The property sheet

When you select an element in the assembly by clicking it, details about the element are displayed in the property sheet. In this pane, you can configure the element's properties. The options available to you in the property sheet are specific to the type of element that you are working with. For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down menu.

You can release the property sheet by clicking the Close icon ✕.

The behavior of an assembly

The assembly runs policies in order and acts on different contexts of the API call. When an API call is made, security and rate limits are enforced before the assembly is executed. During the assembly, the flow can be branched, thrown, and caught according to the policies available. The message context flows through the assembly, being used and altered by various policies. In addition to the message, other contexts can be accessed and created.

Security and rate limiting
: Before the assembly is executed, security and then rate limits are enforced.
First, security definitions and CORS access control are used to authenticate an API call. Any API Key security definitions are used to identify applications that have subscriptions to a Product containing the API. If a security definition does not allow access, the API call is rejected.

: If an application is identified by its client ID or client secret, a rate limit can be enforced based on the Plan or operation called.

The assembly execution sequence
: The assembly is executed in order from the initial, filled, circle to the final, unfilled, circle. However, there is room for branching, when `if` and `operation-switch` logic constructs are used, or for the remaining assembly to be ignored when a `throw` policy is executed.
The message is the context that is acted upon by any policy that isn't otherwise configured. At the beginning of the API call, the message is empty and at the end of the API call, the message is used as the response.

: The request context contains the information that is sent by the API caller and varies with the type of operation called as well as the configuration of that operation. For example, a GET operation can never have a populated `request.body`, and you can configure an operation to have `request.parameters` (query parameters).
The first policy in an assembly acts on the request and produces the first instance of the message. If there are no policies, the request is returned to the caller.

Managing contexts
: Because the message can be overwritten, it is useful to create and reference new contexts where possible so that they are saved and reused during the API call.

  - Use the `map` policy to overwrite the message context when you need to execute a policy that only acts on the message.
  - Use the request context when you want to use the original request made to the API.
  - Use the `map`, `invoke`, and `proxy` policies to create new contexts when you want to save your message.

Note: When you create a new context, unless you are also mapping to the message, the message is overwritten with an empty object.

For example, **invoke** is the first policy in the assembly and its response overwrites the request as the message. The message is then acted upon by a **validate** policy, and a **map** policy then saves the message as a new context, ready for a second **invoke** policy to overwrite the message without losing the first **invoke** policy's output.

You can also access contexts outside of the message or your custom contexts, but these contexts cannot be written to. For a list of contexts, see API Connect context variables.

Branches and catches

Using logic constructs, such as **operation-switch** or **if**, you can execute different sections of the assembly when certain conditions are fulfilled. When the assembly branches, the subsection of the assembly contained by the construct is executed in the same manner as a complete assembly. However, contexts are shared with the complete assembly.

When a catch is triggered, either by an error occurring during the execution of a policy or because a **throw** policy is encountered, the rest of the assembly flow is ignored. All contexts are shared by the catch being executed and when the end of the catch is reached, the API call is completed. There is no way to return from a catch to the rest of the assembly.

Related concepts

- Variable references in API Connect

Related tasks

- Adding elements to your assembly

Related reference

- API Connect context variables

Adding elements to your assembly

Create your API assembly by using the Assemble view in the API Designer. You can access the Assemble view by clicking the Gateway tab in the API editor.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can use the assembly tool in the UI to create assemblies that are used to manipulate requests made to or responses made by any of your API's operations.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

If you want to use the Source view to edit the source of your API definition, see execute OpenAPI extension for more information about how to configure the OpenAPI YAML.

For more information about the use of the assembly tool, see The assembly editor.

Procedure

To add elements to your assembly using the assembly tool, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Use the Policies view to add an element.
   Note: You must specify a version for the policy. To prevent errors due to incompatible versions of a policy between the API and the gateway, API Connect will only publish an API to the gateway if the policy version in the API exactly matches the version present on the gateway.
   a. Select the Gateway tab, then click Policies in the navigation pane.
   b. Click the plus icon ⊕ at the appropriate location on the canvas to view the palette that lists the available elements.
   c. Select the element that you want to add from the palette to the canvas by clicking it.
      Note:
      - Elements are applied in order from the initial, unfilled, circle to the final, filled, circle.
      - Unless an Operation Switch element is used, the whole assembly applies to every operation in the API.
   d. Add and edit the element properties by clicking the element and using the property sheet.
      For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down list. For information about the properties of policy elements, see API policies and logic constructs. For information about the properties of logic constructs, see Logic Constructs.
   e. Specify a version for the policy.
      To add a version, click the Source icon </>, and complete the **version** section of the policy YAML. For example:

```
execute:
  - invoke:
      version: 2.2.0
      title: invoke
...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions. For more information about how to configure the policy OpenAPI YAML, see execute OpenAPI extension.

4. Optional: Repeat Steps 3.b to 3.e for any additional elements that you want to add.
5. Click Save to save your changes.

Results

You have added one or more elements to your assembly.

Handling errors in the assembly

Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

About this task

Note: This task relates to configuring an OpenAPI 2.0 API definition. For details on how to configure an OpenAPI 3.0 API definition, see Editing an OpenAPI 3.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The catch section of the assembly is used to implement an assembly in the instance that an error is thrown during the assembly execution. For example, the assembly could contain a **`throw`** element, the API caller could fail to authenticate, or a policy could fail to execute correctly. Each error can be handled with a different catch and each catch can handle multiple status errors.

Procedure

To create a catch and include elements in it, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to apply a catch to.
3. Select the Gateway tab, then click Policies in the navigation pane.
4. Set the Show catches toggle in the menu bar of the canvas to the Show position.
5. Click Catches in the canvas, or a Catch icon if one is displayed.
   The property sheet for the API's catches opens.
6. To add a default catch that is executed when an otherwise uncaught error is thrown, click Add default catch.
   Note: If you have a default catch before another catch in precedence, the default catch will activate even when the other catch's error is thrown.
7. To add a new catch, click Add catch.
8. To specify which errors the catch applies to, use the Select errors field and select the appropriate error.
9. Optional: To remove an error case from a catch, clear the corresponding checkbox from the Select errors drop-down. To clear all the error cases from a catch, click the cross corresponding to the total number of error cases in the Select errors field.
10. Optional: To change the precedence of your catches, use the Move up or Move down icons.
    If an error case is handled by multiple catches, the catch at the beginning of the list is applied.
11. To add an element to a catch from the palette, click the plus icon at the appropriate location in the assembly flow. Select an element that you want to apply to the catch by clicking it.
12. Click Save to save your changes.

What to do next

If you added a catch for ConnectionError, SOAPError, or OperationError, you must add the same error to the Stop on error setting for the **`Invoke`** policy in your assembly. Otherwise, if the error occurs during the execution of the **`Invoke`** policy, it is not caught, the policy execution is allowed to complete, and the assembly flow continues. For details on configuring an **`Invoke`** policy, see Invoke.

For details of all the errors that can be returned by the assembly and are available to the catch function, see Error cases supported by assembly catches.

Specifying the gateway type for an API definition

An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). A default gateway type is set when you create an API definition, but you can edit the API configuration to specify a different gateway type.

About this task

Note: This task relates only to an OpenAPI 2.0 API definition. An OpenAPI 3.0 API is supported only on the DataPower API Gateway.
DataPower Gateway (v5 compatible) has been available with IBM® API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see API Connect gateway types.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You must specify which type of gateway each API uses. APIs can use only one type of gateway.

When you create a new API definition, or import an API definition that has no gateway type configured, the gateway type defaults according to the gateway types of the gateway services that are enabled in your Sandbox Catalog, as indicated in the following table.

Table 1. Default gateway type for a new API definition

Gateway Service enablement in Sandbox Catalog	Default gateway type for new API definition
No gateway services enabled	DataPower API Gateway
At least one gateway service of each type enabled	DataPower API Gateway
One or more gateway services only of type DataPower API Gateway enabled	DataPower API Gateway
One or more gateway services only of type DataPower Gateway (v5 compatible) enabled	DataPower Gateway (v5 compatible)

If you import an API definition that already has a gateway type configured, that gateway type is retained.
Note that when you modify your API definitions to use a specific gateway type, you must ensure that each policy and policy version in the API are supported by the gateway type. DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

For example, DataPower Gateway (v5 compatible) supports version **1.0.0** of the **invoke** policy, but DataPower API Gateway requires version **2.0.0**.

For information on policies, see execute.

For information on policy versions, see the documentation for each individual policy. For example, to review the **invoke** policy, see invoke.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Click the title of the API you want to update.
3. Select the Gateway tab, then click Gateway and portal settings.
4. Select the type:
   - datapower-gateway for the DataPower Gateway (v5 compatible)
   - datapower-api-gateway for the DataPower API Gateway.

   Note: The Gateway Type selection is not applicable if the Enforced option is disabled, because an unenforced API is not managed on an API Connect gateway.
5. Click Save to retain the changes.
6. In the navigation pane, select Policies to ensure that your API uses policies that are supported by the gateway type you selected.

   Policies in an existing API that are not supported by the new gateway type are grayed out and flagged with a red exclamation point: .

   Some policies are supported by only one gateway type. Some policies are supported by both gateway types, but require a different version of the policy for each gateway type.

Related tasks

- Converting an API definition for deployment to the DataPower API Gateway

Converting an API definition for deployment to the DataPower API Gateway

IBM® API Connect provides two gateway types, DataPower® Gateway (v5 compatible) and DataPower API Gateway. If you have an API definition that was developed for the DataPower Gateway (v5 compatible) and you want to port it to the DataPower API Gateway, you must make changes to convert it before deployment.

About this task

Note: This task relates only to an OpenAPI 2.0 API definition. An OpenAPI 3.0 API is supported only on the DataPower API Gateway.
DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

To convert an API for deployment to the DataPower API Gateway, you must, as a minimum, change the gateway type setting on the API definition, and ensure that each policy and policy version in the API are supported by the DataPower API Gateway; DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

In addition, depending on the specific policies in your API assembly, some policies might require modification to adapt them for use with the DataPower API Gateway.

The following procedure provides instructions for converting your API definition:

Procedure

1. Set the gateway type for the API to DataPower API Gateway, and ensure policy type and version compatibility. For details, see Specifying the gateway type for an API definition.
2. Refer to the DataPower API Gateway porting notes for detailed instructions on how to convert your API definition.

Modifying a GraphQL schema

For a GraphQL API, an editor is provided for modifying the associated GraphQL schema.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Procedure

To open the GraphQL schema editor for a GraphQL API, complete the following steps:

1. Open the API for editing, as described in Editing an OpenAPI 2.0 API definition.
2. Select the GraphQL Schema tab.

What to do next

For full details on how to use the editor, see Using the GraphQL schema editor.

Validating the OpenAPI YAML source

How to validate the OpenAPI YAML source against swagger specifications when you're editing an API.

About this task

When you're configuring an API in the API editor, you can validate the OpenAPI YAML source by clicking Validate > With specifications. If any swagger parser validation errors are found, an error icon is displayed that you can then click to see the individual errors, and where they occur in the YAML. You can then fix the errors, save the updates, and continue to validate the OpenAPI YAML throughout the API configuration process.
Note:

- OpenAPI validation does not show gateway or policy errors that will stop your API from being staged or published. This type of validation occurs only when you try to stage or publish your API. For more information, see Staging an API or Publishing an API.
- You can also validate your API document by using API governance rulesets. For more information, see Validating an API document by using API governance.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Procedure

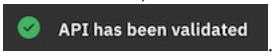
To validate the OpenAPI YAML source, complete the following steps.

1. Open the required API for editing, as described in Editing an OpenAPI 2.0 API definition or Editing an OpenAPI 3.0 API definition.
2. Click Validate > With specifications from the menu bar in the header.

3. If any swagger parser validation errors are found, an error icon is displayed on the header menu bar, with the number of OpenAPI errors found displayed next to the icon.

   For example,

.
4. You can then click the error icon to see a detailed list of the errors, and their location in the YAML. If you're in Form view, you can click an individual error to navigate directly to the relevant API section in the form.
5. After you fix the error, click Save to save your updates, and then click Validate again.
6. When all of the errors are fixed, the validation returns a success message.

   For example, 

.

Related concepts

- Testing an API

Related tasks

- Activating an API
- Staging an API

Validating an API document by using API governance

How to use API governance rulesets to validate and enforce organizational governance policies and best practices. You can validate APIs from the API Manager UI, the API Designer UI, or the toolkit CLI.

Before you begin

Before you can use API governance rulesets to validate your API documents, the API governance optional add-on must be enabled on your management subsystem by your system administrator. See Enabling API governance on Kubernetes, and Enabling API governance on VMware for more information. If API governance is enabled in your deployment, the API governance resource is displayed on the Resources page in the API Manager.

The API governance service is available to all user roles.

Tip: Before using the API governance service to validate your API document, you should run Validate > With specifications to validate the OpenAPI YAML source. For more information, see Validating the OpenAPI YAML source.

About this task

API governance is an IBM® API Connect service that can be used to validate and enforce organizational governance policies and best practices to your API development process. API governance contains the following types of rulesets:

- Provider organization rulesets - these are custom rulesets that contain the rules that are created in, and are specific to, your provider organization.
- Global rulesets - these are preconfigured IBM and Spectral rulesets that contain the rules that are shared with your provider organization, and cannot be edited. Note that the Spectral ruleset names are prefixed by spectral-, and that their version matches the version of that ruleset that's available in Spectral.

You can create your own provider organization rulesets to validate your Swagger, OpenAPI, and AsyncAPI documents against, or use the global rulesets that are provided by default. For more information about configuring rulesets, see Configuring API governance in the API Manager.

API governance in IBM API Connect is based on the open-source Spectral linter; for more information about Spectral, see https://docs.stoplight.io/docs/spectral/674b27b261c3c-overview.

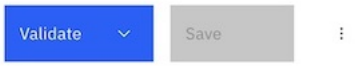
You can complete this task by using the API Manager UI, and by using API Designer when the UI is online, and is connected to a cloud instance that has the API governance microservice enabled on it. You can also use the toolkit CLI; for details see Validating an API document by using the toolkit CLI.

The following methods can be used to validate your API documents with rulesets by using the API Manager or the API Designer UIs:

- From within an API
- From within the Rulesets tab

Procedure

- To validate an API document from within the API, complete the following steps.
  1. Open the required API for validating, as described in Editing an OpenAPI 2.0 API definition or Editing an OpenAPI 3.0 API definition.
  2. Click Validate > With rulesets from the menu bar in the header.

  3. Select the rulesets that you want to validate your API document against, and click Next.
  4. Select the rules from within the rulesets that you want to use for the validation. By default, all of the rules are selected.
  5. Click Validate.
     If any validation errors are found, a scorecard is displayed showing the results of the validation. You can click on an error to take you to the specific section of the API that needs correcting. The API can then be updated to resolve the validation errors, and validated again. If no errors are found, a success message is displayed.
  6. Optional: You can click Download to download the results of the validation into a .csv file.
  7. Optional: You can click Back to change the rulesets and rules, and then run the validation again.
  8. Click Done when finished.
- To validate one or more draft API documents from within the Rulesets tab, complete the following steps.
  1. In the API Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. Click Validate.
  4. Select the rulesets that you want to validate your API document against, and click Next.
  5. Select the rules from within the rulesets that you want to use for the validation. By default, all of the rules are selected.
  6. Select the APIs that you want to validate, and click Validate.
     If any validation errors are found, a scorecard is displayed showing the results of the validation. You can click on an error to take you to the specific section of the API that needs correcting. If no errors are found, a success message is displayed.
  7. Optional: You can click Download to download the results of the validation into a .csv file.
  8. Optional: You can click Back to change the rulesets, rules, and APIs, and then run the validation again.
  9. Click Done when finished.

Results

Your API document is validated against one or more rulesets.

What to do next

You can use the results of the validation to help you improve your API document. You can also use the results to help you edit the rulesets, and create new rulesets; see Configuring API governance in the API Manager for more information.

- **Validating an API document by using the toolkit CLI**
  How to use the API Connect toolkit CLI to run API governance validation scans on your API documents.

Related information

- The API governance ruleset lifecycle

Validating an API document by using the toolkit CLI

How to use the API Connect toolkit CLI to run API governance validation scans on your API documents.

About

API governance is an IBM® API Connect service that can be used to validate and enforce organizational governance policies and best practices to your API development process. The following information provides details of the toolkit CLI commands for running and managing validation scans on your API documents. You can also validate an API document by using the API Manager and the API Designer UIs. For more information about validation scanning, see Validating an API document by using API governance.

The following methods can be used to validate your API documents with rulesets by using the toolkit CLI:

- Validate a single API document that is stored locally
- Validate one or more draft APIs that are stored on the management server

Validate a single API document that is stored locally

To validate a single API document against one or more rulesets, complete the following steps. The OpenAPI document must be stored locally.

1. Log in to the toolkit as the owner or an admin of your provider organization.
   For example:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Run the **compliance:validate** toolkit command in `governance` mode.

   ```
   apic --mode governance compliance:validate --org <target_org> --server <server_url> --rulesets <ruleset_name_1>,
   <ruleset_name_2> API_FILE
   ```

   where:
   - `<target_org>` is the name of the provider organization that owns the APIs to be validated.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `<ruleset_name_1>,<ruleset_name_2>` is a comma-delimited list specifying the names of the rulesets to be applied to the API. (Specify each ruleset name; do not use the ruleset IDs.)
   - `API_FILE` is the path and filename of the OpenAPI document to be validated.

Validate one or more draft APIs that are stored on the management server

The APIs must be stored as draft APIs in your provider organization; this operation validates all of the draft APIs that are listed in the `COMPLIANCE_REQUEST_FILE`.

1. Create a JSON `COMPLIANCE_REQUEST_FILE` that specifies the location of the draft APIs and the rulesets to use during validation.
   The following example shows a configuration file that specifies two draft APIs, and two rulesets. In the `spectral-oas` ruleset, the `info-contact` rule is disabled and will be ignored during validation.

   ```
   {
     "apiUrls" : ["https://example.com/api/orgs/768a5f80-177c-4100-b2b6-57ce6c4bf761/drafts/draft-apis/a08898fe-b422-41d3-
   b058-c3280d632e93","https://example.com/api/orgs/768a5f80-177c-4100-b2b6-57ce6c4bf761/drafts/draft-apis/26692eab-f43b-
   4af3-985c-2deaa401eb7f"]
     "config" : {
       "rulesets": [
         {
           "rulesetName": "spectral-oas",
           "disabled": ["info-contact"]
         },
         {
           "rulesetName": "spectral-owasp",
           "disabled": []
         }
         ]
    }
   }
   ```

   where:

- **`apiUrls`** is a comma-delimited list of URLs. Each URL specifies the location of the draft APIs in a particular provider organization.
- **`rulesetName`** is a list of rulesets to be applied to the APIs during validation.
  For each ruleset in the list, you can optionally add the **`disabled`** property with a comma-delimited list of rules to ignore during validation.

2. Log in to the toolkit as the owner or an admin of your provider organization. See Step 1 for details.
3. Run the **compliance:validate_all** toolkit command in **`governance mode`**.

```
apic --mode governance compliance:validate_all --org <target_org> --server <server_url> COMPLIANCE_REQUEST_FILE
```

where:

- ***`<target_org>`*** is the name of the provider organization that owns catalog containing the APIs to be validated.
- ***`<server_url>`*** is the management server URL that you use for logging in to the toolkit.
- ***`COMPLIANCE_REQUEST_FILE`*** is the path and filename of the configuration file that lists the APIs and rulesets to be used during the validation (for example, validate-apis.json).

Editing an OpenAPI 3.0 API definition

IBM® API Connect provides a form based editor that enables you to configure APIs that conform to the OpenAPI 3.0 specification.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Procedure

You can open an API for editing in either of the following ways:

- During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
- To open an existing API for editing, complete the following steps:
  1. In the navigation pane, click Develop, then select the APIs tab.
  2. Click the title of the API version that you want to work with.

- **OpenAPI 3.0 support in IBM API Connect**
  IBM API Connect supports the OpenAPI 3.0 specification, with some limitations. The current implementation includes complete support for the Berlin Group NextGen PSD2 requirements.
- **Specifying API metadata**
  API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.
- **Defining servers for an API**
  Server definitions in an API provide information for connecting to target servers.
- **Enforcing security requirements on an API**
  To enforce security requirements on an API, you apply previously created security scheme components that define various aspects of API security configuration.
- **Specifying external documentation for an API**
  You can reference an external resource for extended documentation for an API.
- **Adding tags to an API**
  You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- **Defining Paths for an API**
  A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.
- **Defining components for an API**
  Components are reusable objects relating to various aspects of your API definition. A component has no effect on the API unless it is explicitly referenced from elsewhere in your API definition.
- **Specifying gateway and portal settings**
  Define general configuration settings for your API.
- **Defining target services for an API**
  A target service defines a web service that you want to use in your API definition.
- **Setting API properties**
  In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.
- **Configuring application authentication for an API**
  Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.
- **Configuring activity logging**
  You can configure your logging preferences for the API activity that is stored in analytics, overriding the default activity logging behavior.
- **Including elements in your API assembly**
  An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.
- **Validating the OpenAPI YAML source**
  How to validate the OpenAPI YAML source against swagger specifications when you're editing an API.
- **Validating an API document by using API governance**
  How to use API governance rulesets to validate and enforce organizational governance policies and best practices. You can validate APIs from the API Manager UI,

the API Designer UI, or the toolkit CLI.

OpenAPI 3.0 support in IBM API Connect

IBM® API Connect supports the OpenAPI 3.0 specification, with some limitations. The current implementation includes complete support for the Berlin Group NextGen PSD2 requirements.

Overview

A Product can contain any combination of OpenAPI 2.0 and OpenAPI 3.0 APIs. When you publish a Product that contains an OpenAPI 3.0 API, that API is validated to ensure that it is syntactically correct, and that references to configuration resources and policies resolve correctly, in the same way that OpenAPI 2.0 APIs are validated.

You can also validate OpenAPI 3.0 APIs in your local file system by using the **`apic validate`** command provided by the developer toolkit CLI; for details, see Validating the YAML or JSON definition of an API or Product.

If you retrieve an API object by using the developer toolkit CLI or the API Connect REST APIs, there is an **`oai_version`** property that defines which OpenAPI version the API represents.

There is no OpenAPI 3.0 API support with the DataPower® Gateway (v5 compatible); OpenAPI 3.0 API support is provided by the DataPower API Gateway **only**.

Limitations

The limitations to the OpenAPI 3.0 support in IBM API Connect are as follows:

User interface limitations

- Some aspects of the OpenAPI 3.0 specification are not currently supported by the user interface. In such circumstances, use the OpenAPI source directly.
- Validation errors that are identified locally are reflected in the API editor header almost immediately. However, some validation errors can be identified only by the API Manager backend and are not reflected until the API is saved; this means that, on saving an API, some validation errors might appear or disappear.
- The user interface does not currently support referencing a request body component from the request body of an operation. If you want to reference a request body component, add the reference to the request body of the operation directly in the OpenAPI source:

  ```
  requestBody:
    $ref: '#/components/requestBodies/request_body_component_name'
  ```

  The reference **is** confirmed on the details page for the request body in the user interface however.
- If, in the API Designer user interface, you create and activate an API of the same name and version as one that has already been activated from the API Manager user interface, the Edit Rate Limit operation fails.

Limitations for APIs that are enforced by the DataPower API Gateway

To suppress the error message for the following limitations and publish the API, set the **`x-ibm-configuration.compatibility.suppress-limitation-errors`** property to **`true`** as shown in the following example:

```
compatibility:
  suppress-limitation-errors: true
```

Attention: Enabling **`suppress-limitation-errors`** merely disables the error messages so that the API can be published.

- General limitations.
  - The **`servers`** array cannot contain more than one server.
  - The **`url`** entry in the **`servers`** array cannot contain **`variables`**.
  - The **`path`** objects cannot contain server object overrides.
  - There is no support for converting a WSDL defined SOAP service into an OpenAPI 3.0 API.
  - There is no support for callbacks or links, and APIs containing them will not be published.
  - The following constructs of Open API 3.0 are less commonly used, and are not supported:
    - **`allowEmptyValue`**
    - **`allowReserved`**
- Assembly policies.
  - All policies are supported.
  - JSON schema support is limited to Draft 4 and earlier. Therefore, the following policies are limited to those drafts:
    - **`gatewayscript`**
    - **`set-variable`**
    - **`switch`**
    - **`json-to-xml`**
    - **`xml-to-json`**
    - **`parse`**
    - **`xslt`**
    - **`map`**
    - **`validate`**

    For detailed information about the Draft 4 and earlier specifications, see https://json-schema.org/specification-links.html#draft-4.
  - Security.
    - The OpenID Connect (OIDC) security scheme is not supported.
    - The use of cookies in an API key is not supported.

Specifying API metadata

API metadata provides summary information about the API. You enter API metadata in the Info section of the API definition.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand the General section, then click Info.
3. Provide the following information:
   - Title (required): The title of the API.
   - Name: The name was generated automatically when the API was created and cannot be changed. The value in the Name field is a single string that is used to identify the API in developer toolkit CLI commands. To view the CLI commands to manage draft APIs, see apic draft-apis.
   - Version: The API version is specified when the API is initially created and cannot be changed thereafter. You can, however, create a new version of an existing API; see Creating a new version of an API definition.
   - Description: An optional description of the API. You can use CommonMark syntax for rich text representation.
   - Terms of Service: A URL to the Terms of Service for the API.
   - Contact Name: The identifying name of the contact person/organization.
   - Contact URL: A URL pointing to the contact information.
   - Contact Email: The email address of the contact person/organization.
   - License Name: The license name used for the API.
   - License URL: A URL to the license used for the API.
4. Click Save when done.

Defining servers for an API

Server definitions in an API provide information for connecting to target servers.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand General then, if there are already one or more servers defined, expand Servers.
3. To create a new server definition, click the add icon ⊕ alongside Servers in the navigation pane. To edit an existing server definition, click the server URL in the navigation pane.
4. Provide the following information:
   - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:

     ```
     /my_basepath
     ```

     For full details on how the server URL is used to determine the full URL endpoint, see Configuring vanity endpoints for a Catalog.
   - Server Description: An optional description of the host designated by the URL. You can use CommonMark syntax for rich text representation.
   - Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
     - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
     - Provide the following information:
       - Server Variable Name (required).
       - Default value (required).
       - An optional rich text description. You can use CommonMark syntax for rich text representation.

- One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add an new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
- If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
- If required, use the breadcrumb trail to return to your server definition for further editing.

5. If you are creating a new server definition, click Create.
   The server definition details are displayed for further editing as described in step 4.
6. Click Save when done.

Enforcing security requirements on an API

To enforce security requirements on an API, you apply previously created security scheme components that define various aspects of API security configuration.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security scheme components, see Defining security scheme components.

The following restrictions exist when you apply security schemes to an API:

- You cannot apply more than two API key security schemes to an API.
- If you apply an API key security scheme for client secret, you must also apply an API key security scheme for client ID.
- If you require the application developer to supply both client ID and client secret, you must apply two separate API key security schemes.
- You can have at most one API key scheme of type client ID, regardless of whether the client ID is sent in the request header or as a query parameter.
- You can have at most one API key scheme of type client secret, regardless of whether the client secret is sent in the request header or as a query parameter.
- You cannot apply more than one basic security scheme to an API. If you apply a basic security scheme, you cannot also apply an OAuth security scheme.
- You can apply at most one OAuth security scheme to an API.

A security requirement specifies one or more security scheme components whose conditions must all be satisfied for the API to be called successfully. You can define multiple security requirements; in this case, an application can call your API if it satisfies any of the security requirements you have defined.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand General.
3. To create a new security requirement for the API, complete the following steps:
   a. Click the add icon ⊕ alongside Security in the navigation pane.
   b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined in security scheme components; see Defining security scheme components.
   If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme component; for more information, see Defining OAuth2 security scheme components.

   If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see Configuring scopes for a native OAuth provider.

   Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the **`X-IBM-Client-Id`** or **`client_id`** must be included in the security credentials so that the correct Plan configuration settings can be enforced.
   c. Click Create. The security scheme selections are shown; you can change them again before saving.
   d. Click Submit when done.
4. To modify an existing security requirement, complete the following steps:
   a. Click Security in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
   b. To change the security schemes for a security requirement, click the edit icon alongside the required security requirement, then change your security requirement selections as required.
   c. Click Submit when done, then click Save.
   d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
   e. To disable security for the API, clear the Require one of the following Security Requirements check box, then click Save.

What to do next

Fore more information on LDAP and Authentication URL, see LDAP authentication and Authentication URL user registry.

- **Enabling CORS support for an API**
  You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

Enabling CORS support for an API

You can enable cross-origin resource sharing (CORS) support for your API. CORS allows embedded scripts in a web page to call the API across domain boundaries.

Before you begin

This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.

About this task

Note:

- CORS support is available only on the DataPower® API Gateway.
- When CORS is enabled, the API Gateway runs the `cors` preflow policy to handle all CORS requests that are made to the API.
- When CORS is enabled and a preflight request is received, only the following API actions are performed:
  - The `cors` preflow policy configures the appropriate response headers.
  - The response headers are set.
- When a preflight request is received, the `request.attributes.isCORSPreflight` flag is set to `true`.
- For all preflight requests, the `security` and `client-identification` preflow policies are always skipped, whether CORS is enabled or not enabled.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

To enable CORS support for an API, complete the following steps:

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings, then click CORS.
   The CORS page opens.
3. Select Enable CORS.
4. Optional: Configure a CORS policy.
   Should you create a CORS policy? Review the following considerations:
   - A CORS policy is an optional part of an API definition. If an API definition has no CORS policy but CORS is enabled, then CORS requests will be accepted from all Origins.
     If you want to accept CORS requests from all Origins, then enable CORS but do not add a CORS policy to the API definition.
   - If you create a CORS policy, then CORS requests will only be accepted from Origins that are explicitly listed in the CORS rules contained in the CORS policy. CORS requests from any other Origins will be rejected.
     If you want to accept CORS requests from only a limited number of Origins (and might also want to configure the Access-Control-Allow-Credentials and Access-Control-Expose-Headers response headers), then you should enable CORS and create a CORS policy. Only those Origins explicitly listed in the allow-origin field of the CORS rules in the CORS policy will be accepted; CORS requests from any Origins not listed will be rejected.
   - To configure a new CORS policy, complete the following steps:
     - Alongside CORS Policy, click Add.
     - To include the header `Access-Control-Allow-Credentials: true` in a response. select Allow Credentials. The `Access-Control-Allow-Credentials` response header tells browsers whether to expose the response to frontend JavaScript code when the request's credentials mode (`Request.credentials`) is set to `include`.
     - To append one or more of the following values to the `Access-Control-Expose-Headers` response header, select Expose headers, and select from the following options:
       - Predefined - The predefined value of the Gateway. This option is selected by default.
       - Backend - The value of `Access-Control-Expose-Headers` from the backend response.
       - Custom - A custom string.
     - Click Create.
     - Alongside Allowed Origins click Add.
     - Enter the origin URL, then click Create. This setting indicates that the response can be shared with requesting code from the specified origin.
   - To modify an existing CORS policy, click its Allowed Origins entry on the CORS page. You can then modify individual origin URLs, add further origins, and change the Allow Credentials setting.

   Example

   The following example has three rules:
   - Accept an Origin header of `https://example.com`, and return `Access-Control-Allow-Credentials: true` in the CORS response.
   - Accept an Origin header of `http://domain.com`. No `Access-Control-Allow-Credentials` header will be returned in the response.
   - Accept any of the following Origin headers:
     - `http://example2.com`
     - `http://example3.com`
     - `http://example4.com`

     No `Access-Control-Allow-Credentials` header will be returned in the response.

```
cors:
  enabled: true
  policy:
    -
      allow-origin:
        - 'https://example.com'
      allow-credentials: true
    -
      allow-origin:
```

```
          - 'http://domain.com'
      -
        allow-origin:
          - 'http://example2.com'
          - 'http://example3.com'
          - 'http://example4.com'
```

5. Click Save to save your changes.
6. Optional: To implement your own CORS solution using custom OPTIONS operations, complete the following steps:
   a. Add the following headers to your HTTP responses:

      ```
      Access-Control-Allow-Origin: https://<portalhostname>
      Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
      ```

      Where `<portalhostname>` is your Developer Portal host name.
   b. Optional: You can proxy your API through API Connect as an enforced invoke API so that CORS is handled automatically.

Important:

- If you implement your own CORS solution, you **must** disable the CORS option described in step 3
- CORS preflight requests are sent by using the HTTP **`OPTIONS`** method. Therefore, if you require these requests to be handled by the API Connect gateway then you must enable the **`OPTIONS`** method for all APIs that will handle preflight requests; see Defining Paths for a REST API.
- **`OPTIONS`** requests are counted as API calls against any configured rate limit. Note that you can apply rate limits to individual operations; see Defining rate limits for an API operation.

Specifying external documentation for an API

You can reference an external resource for extended documentation for an API.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand the General section, then click External Documentation.
3. Provide the following information:
   - URL (required): The URL for the target documentation.
   - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
4. Click Save when done.

Adding tags to an API

You can include tags in API definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand the General then, if there already one or more tags defined, expand Tags.
3. To add a new tag, click the add icon ⊕ alongside Tags in the navigation pane. To edit an existing tag, click the tag name in the navigation pane.
4. Provide the following information:
   - Name (required): The name of the tag.
   - Description: An optional description of the tag. You can use CommonMark syntax for rich text representation.
   - External Documentation > URL (available when editing an existing tag): The URL for target documentation that describes this tag.

- External Documentation > Description (available when editing an existing tag): An optional description of the target documentation.

5. If you are creating a new tag, click Create.
   The tag details are displayed for further editing as described in step 4.
6. Click Save when done.

Defining Paths for an API

A Path is a unit of a REST API that you can call. A Path comprises an HTTP verb and a URL path. By configuring the Path, you define how the API is exposed to your developers.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The Path is appended to the server URL to form the full URL. For details on configuring servers, see Defining servers for an API.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. If there are already one or more Paths defined, expand Paths.
3. To create a new Path, click the add icon ⊕ alongside Paths in the navigation pane. To edit an existing Path, click the Path in the navigation pane.
4. Provide the following information:
   - Path: A relative path to an individual endpoint.
     The path is appended to the base path to construct the full URI to access the APIs. The path must start, but not end, with the / character. A parameter at the end of the path can contain a qualifier to match one or more path levels.

     If you specify just the name of the parameter, then one level of that path is matched. If you want to allow for multiple levels of the path, you can prefix the parameter with one of the following qualifiers:

     - * to indicate 0 or more occurrences
     - + to indicate 1 or more occurrences

     The + and * qualifiers can only be used at the end of the path.

     For example, the path:

     ```
     /petstore/{type}/{*category}
     ```

     matches the following paths, where only one type value is matched, but all (0 or more) categories are matched:

     ```
     /petstore/cats
     /petstore/cats/supplies
     /petstore/cats/supplies/health
     /petstore/cats/supplies/health/medicines
     /petstore/cats/supplies/health/medicines/a/b/c
     ```

   - Ref (available when editing an existing Path): A reference to a Path that is defined elsewhere, either in this OpenAPI definition, or in an external file. A reference enables reuse of a set of operations already defined for a Path in this API, or in a file that is either a complete API definition or an OpenAPI fragment.
     The reference uses JSON pointer notation. For example, a reference to a Path called `/mypath` in this API would have the following format:

     ```
     #/paths/~1mypath
     ```

     The `~1` characters are used to escape the `/` character, which is being used literally in the path name rather than as a special character.
     A reference to a path in an external file called path_fragments.yaml would have the following format:

     ```
     file://path_fragments.yaml/paths/~1mypath
     ```

   - Summary: An optional summary, intended to apply to all operations in this path.
   - Description: An optional description, intended to apply to all operations in this path. You can use CommonMark syntax for rich text representation.
5. If you are creating a new Path, click Create.
   The Path details are displayed for further editing as described in step 4.
6. Click Save when done.

- **Defining servers for a Path**
  Server definitions for a Path provide alternative target servers when calling the API at that Path.
- **Defining parameters for a Path**
  Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.
- **Defining operations for a Path**
  Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

Defining servers for a Path

Server definitions for a Path provide alternative target servers when calling the API at that Path.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note: It is unlikely that you will need to define servers for a Path because it applies only if the API operations are distributed across different gateway endpoints, and such a configuration is possible only if you are hosting your API operations on your own runtime endpoints outside of API Connect. For an API that is published to the DataPower API Gateway, all API operations are invoked on the same gateway service.
A server defined for a Path overrides any server defined for the parent API. You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths.
3. Expand the required Path, then, if there already one or more servers defined for the Path, expand Servers.
4. To create a new server definition for the Path, click the add icon ⊕ alongside Servers under the Path in the navigation pane. To edit an existing server definition, click the server URL under the Path in the navigation pane.
5. Provide the following information:
   - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:

     ```
     /my_basepath
     ```

     For full details on how the server URL is used to determine the full URL endpoint, see Configuring vanity endpoints for a Catalog.
   - Server Description: An optional description of the host designated by the URL. You can use CommonMark syntax for rich text representation.
   - Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
     - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
     - Provide the following information:
       - Server Variable Name (required).
       - Default value (required).
       - An optional rich text description. You can use CommonMark syntax for rich text representation.
       - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add an new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
     - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
     - If required, use the breadcrumb trail to return to your server definition for further editing.
6. If you are creating a new server definition, click Create.
   The path details are displayed for further editing as described in 5.
7. Click Save when done.

Defining parameters for a Path

Path parameters can specify variable elements of a URL path, query parameters, headers, or a request body. The parameters apply to all the operations defined under the path. A Path parameter can be overridden at the operation level but cannot be removed there.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to `header`.
- A parameter can have various format styles, the style of a header is always `simple`.

For more information on headers, see Creating a header.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths.
3. Expand the required Path, then, if there already one or more parameters defined, expand Parameters.
4. You can either create a new parameter, or edit any existing parameter.
   - To create a parameter, click the add icon alongside Parameters, then refer to Creating a parameter.
   - To edit an existing parameter, click the parameter name in the navigation pane, then refer to Editing a parameter.

Defining operations for a Path

Operations on a Path defines the mechanisms available for interacting with the API when it is called using that Path. An operation can be of various pre-defined types; for example, a POST operation is most commonly used for creating new resources, while a GET operation is used to retrieve a resource.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths.
3. Expand the required Path, then, if there are one or more operations, expand Operations.
4. To create an operation, click the add icon alongside Operations under the Path in the navigation pane. To edit an existing operation, click the operation name under the Path in the navigation pane.
5. Provide the following information:
   - Verb: Select the operation type from the following options:
     - get
     - post
     - put
     - delete
     - options
     - head
     - patch
     - trace
   - Summary: A summary of what the operation does.
   - Description: A description of the operation behavior. You can use CommonMark syntax for rich text representation.
   - Operation ID: Unique string used to identify the operation. The id must be unique among all operations described in the API.
   - Deprecated: Declares this operation to be deprecated. Consumers should refrain from using the declared operation.
6. If you are creating a Path operation, click Create.
7. Click Save when done.

- **Adding tags to an operation**
  You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.
- **Specifying external documentation for an operation**
  You can referencing an external resource for extended documentation for an API operation.
- **Defining parameters for an operation**
  Operation parameters can, for the URL path associated with that specific operation, define variable elements of the URL path, query parameters, headers, or body parameters.
- **Defining the request body for an operation**
  A request body defines the structure of the body of an API request.
- **Defining responses for an operation**
  Responses define the HTTP status code and data returned in a response body and headers.
- **Enforcing security requirements on an operation**
  To enforce security requirements on an API operation, you apply previously created security scheme components that define various aspects of API security configuration.
- **Defining servers for an operation**
  Server definitions for an API operation provide alternative target servers when calling that operation.

Adding tags to an operation

You can include tags in API operation definition. Such tags are added to the OpenAPI definition of the API but are not used by API Connect for any indexing.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To add a new operation tag, click the add icon ⊕ alongside the Tags entry for the operation in the navigation pane. To edit an existing tag, expand Tags then click the tag name in the navigation pane.
5. Provide the following information:
   - Name (required): The name of the tag.
6. If you are creating a new operation tag, click Create.
   The tag details are displayed for further editing as described in step 5.
7. Click Save when done.

Specifying external documentation for an operation

You can referencing an external resource for extended documentation for an API operation.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click External Documentation.
4. Provide the following information:
   - URL (required): The URL for the target documentation.
   - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
5. Click Save when done.

Defining parameters for an operation

Operation parameters can, for the URL path associated with that specific operation, define variable elements of the URL path, query parameters, headers, or body parameters.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An operation parameter overrides a parameter of the same name defined on the parent Path; see Defining parameters for a Path.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

For more information on headers, see Creating a header.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. You can either create a new operation parameter, or edit any existing parameter.
   - To create an operation parameter, click the add icon alongside Parameter, then refer to Creating a parameter.
   - To edit an existing operation parameter, click the parameter name in the navigation pane, then refer to Editing a parameter.

Defining the request body for an operation

A request body defines the structure of the body of an API request.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Note: The user interface doesn't currently support referencing a request body component from an operation. However, you can add the reference directly to the OpenAPI YAML source for your API definition, for example:

```
paths:
  /mypath:
    get:
          .
          .
          .
      requestBody:
        $ref: '#/components/requestBodies/my_requestbody_component'
          .
          .
          .
```

For information on request body components, see Defining request body components.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then click requestBody.
4. Provide the following information:
   - Description: An optional description of the request body. This could contain examples of use. You can use CommonMark syntax for rich text representation.
   - Content: A content definition describes the content of the request body.
     To define a new content definition for the request body, click Add, then refer to Creating a content definition.

     To edit an existing content definition, click the content definition name, then refer to Editing a content definition.
   - Required: Determines if the request body is required in the request.
5. Click Save when done.

Defining responses for an operation

Responses define the HTTP status code and data returned in a response body and headers.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there are any existing responses, expand Responses.
4. You can either create a new response, or edit any existing response.
   - To create a response, click the add icon ⊕ alongside Responses, then refer to Creating a response.
   - To edit an existing response, click the response name in the navigation pane, then refer to Editing a response.

Enforcing security requirements on an operation

To enforce security requirements on an API operation, you apply previously created security scheme components that define various aspects of API security configuration.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

For details on how to create and configure security scheme components, see Defining security scheme components.

A security requirement specifies one or more security scheme components whose conditions must all be satisfied for the API operation to be called successfully. You can define multiple security requirements; in this case, an application can call your API operation if it satisfies any of the security requirements you have defined.

Any security requirements that you define for an operation completely override any security requirements defined on the parent API. If you do not define any security requirements for an operation, or you delete all security requirements from an operation, the operation inherits the security requirements defined on the parent API. For more information, see Enforcing security requirements on an API.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, then expand the required operation.
4. To create a new security requirement for the operation, complete the following steps:
   a. Click the add icon ⊕ alongside the Security Requirements entry for the operation in the navigation pane.
   b. Select the security schemes that you want to include in this security requirement. The security schemes listed are those that have been defined in security scheme components; see Defining security scheme components.
   If a selected security scheme is of type OAuth2, select the required scopes; the scopes available for selection are those that were specified in the security scheme component; for more information, see Defining OAuth2 security scheme components.

   If you are applying the OAuth2 security scheme to an API that is enforced by the DataPower API Gateway, you only need select any scopes if Advanced scope check after token generation is not enabled in the native OAuth provider associated with the security scheme. If a default scope has been set in the native OAuth provider and the API request doesn't contain any scope, the default scope is used; for more information, see Configuring scopes for a native OAuth provider.

   Note: The following additional requirement applies to security schemes that will be used with an OAuth third party provider. If you select an OAuth security scheme for protecting a consumer API, you must also include an API key security scheme, as the **`X-IBM-Client-Id`** or **`client_id`** must be included in the security credentials so that the correct Plan configuration settings can be enforced.
   c. Click Create. The security scheme selections are shown; you can change them again before saving.
   d. Click Submit when done.
5. To modify an existing security requirement, complete the following steps:
   a. Click the Security entry for the operation in the navigation pane. All previously defined security requirements are listed; the security schemes included in each security requirement are shown.
   b. To change the security schemes for a security requirement, click the edit icon alongside the required security requirement, then change your security requirement selections as required.
   c. Click Submit when done, then click Save.
   d. To delete a security requirement, click the appropriate delete icon , click Delete to confirm, then click Save.
   e. To disable security for the operation, clear the Require one of the following Security Requirements check box, then click Save.

Note: These settings completely override any security requirements defined on the parent API; see Enforcing security requirements on an API.

Defining servers for an operation

Server definitions for an API operation provide alternative target servers when calling that operation.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A server defined for an operation overrides any server defined for the parent API or Path. You can define more than one server but only the first is used by API Connect.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Paths, then expand the required Path.
3. Expand Operations, expand the required operation, then, if there already one or more servers defined for the operation, expand Servers.
4. To create a new server definition for the operation, click the add icon ⊕ alongside Servers under the operation in the navigation pane. To edit an existing server definition, click the server URL under the operation in the navigation pane.
5. Provide the following information:
   - Server URL (required): The specified URL is used to determine the full URL endpoint for the calling the API, taking into account any vanity endpoint configuration in the Catalog in which the API is published. For an API that is enforced by the DataPower API Gateway, the value entered here is interpreted as the basepath so you would typically provide only the basepath value; for example:

     ```
     /my_basepath
     ```

     For full details on how the server URL is used to determine the full URL endpoint, see Configuring vanity endpoints for a Catalog.
   - Server Description: An optional description of the host designated by the URL. You can use CommonMark syntax for rich text representation.
   - Server Variables (available when editing an existing server definition): A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
     - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
     - Provide the following information:
       - Server Variable Name (required).
       - Default value (required).
       - An optional rich text description. You can use CommonMark syntax for rich text representation.
       - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add an new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
     - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
     - If required, use the breadcrumb trail to return to your server definition for further editing.
6. If you are creating a new server definition, click Create.
   The path details are displayed for further editing as described in 5.
7. Click Save when done.

Defining components for an API

Components are reusable objects relating to various aspects of your API definition. A component has no effect on the API unless it is explicitly referenced from elsewhere in your API definition.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The following subtopics detail the various types of API component:

- **Defining schema components**
  Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.
- **Defining response components**
  Response components define reusable response objects that specify the HTTP status code and data returned in a response body and headers.
- **Defining parameter components**
  Parameter components define reusable parameters that can specify variable elements of a URL path, query parameters, headers, or a request body.
- **Defining example components**
  An example component is a reusable example that you can add to an API to make the specification of your API clearer.

- **Defining request body components**
  A request body component is a reusable definition of the structure of the body of an API request.
- **Defining header components**
  A header component defines a reusable custom header that can be sent in an API request or returned in an API response.
- **Defining security scheme components**
  A security scheme component specifies all the settings for a particular aspect of API security; for example, the user registry that you use to authenticate access to the API.
- **Defining link components**
  A link component is a reusable link definition that can be used to describe how various values returned by one operation can be used as input for other operations. In this way, links provide a known relationship and traversal mechanism between the operations.

Defining schema components

Schema components define reusable schemas that provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

An API created in OpenAPI 3.0 includes a section where API payload definitions are defined. The payload definitions describe the structure of data transmitted in API requests and responses, utilizing a specialized JSON schema variant specific to OpenAPI. Each payload definition corresponds to a schema, which lists the expected data fields. These data fields are represented as properties within the schema and include a set of attributes that describe their type, permissible values, mandatory status, and other characteristics. Although numerous attributes are available, the majority of them are infrequently utilized.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a schema component, you reference it from elsewhere in your API definition. You can reference a schema from the following locations:

- An API Path parameter; see Defining parameters for a Path.
- An API operation parameter; see Defining parameters for an operation.
- The request body of an API operation; see Defining the request body for an operation.
- An API operation response; see Defining responses for an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more schemas components defined, expand Schemas.
3. You can create a new schema component, or edit any existing schema component.
   - To create a schema component, click the add icon alongside Schemas, then refer to Creating a schema component.
   - To edit an existing schema component, click the schema component name in the navigation pane, then refer to Editing a schema component.

- **Creating a schema component**
  Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.
- **Editing a schema component**
  Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

Creating a schema component

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can create schemas in various places in your API definition.

Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.
Launch the schema creation window. For details of the areas in your API definition from where you can create a schema, see the following topics:

- Editing a content definition
- Editing a header
- Editing a parameter
- Defining schema components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. To create a schema definition, open the Componentselement and click the add icon ⊕ alongside Schema.
   The Add object wizard for a schema is displayed.
2. Provide the following information:
   -
     - Schema Name: If you are creating a schema component, this name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/schemas/Name`

     - Title: The schema title.
     - Type: The schema data type; select one of the following:
       - array
       - boolean
       - integer
       - number
       - object
       - string
3. Click Create.
   The schema details are displayed for further editing; see Editing a schema component.

Editing a schema component

Schemas provide developers with information about the request they should make, or the response they should expect to receive, when calling an API operation. You can edit schemas that have been previously created in various places in your API definition.

Before you begin

Note: Schemas are compiled before they are used for validation. Because the compilation process is longer than the validation process, the compiled schema artifacts are stored in a cache. The limited capacity of the cache can cause older entries to be evicted from the cache when newer entries are added. Schemas whose artifacts have been evicted from the cache must be recompiled, which can cause significant delays in validation.
Open the details form for a schema. For details of the areas in your API definition from where you can edit a schema, see the following topics:

- Editing a content definition
- Editing a header
- Editing a parameter
- Defining schema components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The table view presents the properties within a schema alongside their essential attributes, such as name, type, description, and required status. The secondary pop-up enables access for viewing and editing the more detailed and less commonly used attributes that are associated with each data field.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

You can configure the following schema settings:

- Schema Name: Available if you are editing a schema component, this name defines a key that enables this schema to be referenced from elsewhere in the API definition; the reference has the following format:

  `#/components/schemas/Name`

To change the name, click the Source icon, change the value in the field, then click Save
- Optional: Title: The schema title.
- Optional: Type: The schema data type; select one of the following:
  -
    - array
    - boolean
    - integer
    - number
    - object
    - string

  Note: If you change the type of an existing schema, any existing settings that are not relevant to the new type are still retained in the OpenAPI source even though they are no longer displayed in the user interface. You should therefore avoid porting a schema from one type to another but instead either create a new schema or modify the OpenAPI source directly; to open the source editor, click the Source icon </> .
- Properties: The schema property. To add a property to the schema, click Add alongside the Properties section heading, complete the details and click Save.
- Additional schema settings dependent on the selected Type of property. To view and edit these schema settings, click the Options menu next to the property you want to view and select Details.
  - boolean: no type specific settings.
  - integer:
    - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
    - Multiple Of: Division of an integer instance by this value must result in an integer.
    - Maximum: An integer instance must be less than or equal to this value.
    - Exclusive Maximum: An integer instance must be strictly less than this value.
    - Minimum: An integer instance must be greater than or equal to this value.
    - Exclusive Minimum: An integer instance must be strictly greater than this value.
  - number:
    - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
    - Multiple Of: Division of a number instance by this value must result in an integer.
    - Maximum: A number instance must be less than or equal to this value.
    - Exclusive Maximum: A number instance must be strictly less than this value.
    - Minimum: A number instance must be greater than or equal to this value.
    - Exclusive Minimum: A number instance must be strictly greater than this value.
  - object:
    - Max Properties: The number of properties in an object instance must be less than or equal to this value.
    - Min Properties: The number of properties in an object instance must be greater than or equal to this value.
    - Required Properties: An array of property names; every property name in the array must be the name of a property in an object instance. To add a required property name, click Add alongside the Required Properties section heading, enter the name, then click Create.
  - string:
    - Format: An optional data type modifier. For more information, see Data Types in the OpenAPI Specification.
    - Pattern: A regular expression; a string instance must successfully match the expression.
    - Max Length: The maximum allowed number of characters in a string instance.
    - Min Length: The minimum allowed number of characters in a string instance.
    - Enum: An array of string values; a string instance must match one of the array values. To add an enum value, click Add, enter the value, then click Create.
- General settings; select the required options:
  - Required: A property associated with this schema is required.
  - Read Only: A property associated with this schema can be returned in a response but should not be included in a request. For example, properties whose values are system generated identifiers will be defined as Read Only because application clients wouldn't be able to set these.
- Example Value: An example value. Whatever you enter here is displayed as-is in the Developer Portal.
- External Documentation: An external resource for extended documentation; provide the following information:
  - URL (required): The URL for the target documentation.
  - Description: An optional description of the target documentation. You can use CommonMark syntax for rich text representation.
- Xml: Metadata that allows for more fine-tuned XML model definitions; provide the following information:
  - Name: Replaces the name of the element or attribute used for the described schema property.
  - Namespace: The URL of the namespace definition.
  - Prefix: The prefix to be used for the Name.
  - Attribute: Specifies whether the property definition translates to an attribute instead of an element.
  - Wrapped: For an array definition, Specifies whether the array is wrapped (for example, **<books><book/><book/></books>**) or unwrapped (**<book/> <book/>**).
- Click Save when done.

Example

By defining properties, you can create more complex data structures. For example, you could define an "Address" schema that has the following properties:

PROPERTY NAME	PROPERTY TYPE
street1	string
street2	string
city	string
state	string
zip_code	string

and then define a "BankBranch" schema that has the following properties, one of which is a reference to the Address schema:

PROPERTY NAME	PROPERTY TYPE
address	address
type	string
id	string

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining response components

Response components define reusable response objects that specify the HTTP status code and data returned in a response body and headers.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a response component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more response components defined, expand Responses.
3. You can create a new response component, or edit any existing response component.
   - To create a response component, click the add icon ⊕ alongside Responses, then refer to Creating a response.
   - To edit an existing response component, click the response component name in the navigation pane, then refer to Editing a response.

- **Creating a response**
  Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.
- **Editing a response**
  Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.
- **Creating a content definition**
  A content definition describes the content of an API request, response, or parameter. You can create content definitions in various places in your API definition.
- **Editing a content definition**
  A content definition describes the content of an API request, response, or parameter. You can edit content definitions that have been previously created in various places in your API definition.
- **Creating an encoding definition**
  An encoding definition defines a map between a specific schema property and its encoding information.
- **Editing an encoding definition**
  An encoding definition defines a map between a specific schema property and its encoding information

Creating a response

Responses define the HTTP status code and data returned in a response body and headers. You can create responses for the Path operations in your API definition.

Before you begin

Launch the response creation window. For details of the areas in your API definition from where you can create a response, see the following topics:

- Defining responses for an operation
- Defining response components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the response; you can either create a response from scratch, or you can reference a response that is defined in a response component.
   - To create the response from scratch, provide the following information on the Definition tab:
     - Response name: A name for the response.
       If you are creating an operation response, this name is the HTTP status code.

If you are creating a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:

`#/components/responses/Name`

- Description (required): A description of the response. You can use CommonMark syntax for rich text representation.

- To reference a response that is defined in a response component, provide the following information on the Reference tab:
  - Response name: A name for the response.
    If you are creating an operation response, this name is the HTTP status code.

    If you are creating a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:

    `#/components/responses/Name`

  - Reference from: Select a reference to a response component.

2. Click Create.
   The response details are displayed for further editing; see Editing a response.

Editing a response

Responses define the HTTP status code and data returned in a response body and headers. You can edit responses that have been previously created for Path operations in your API definition.

Before you begin

Open the details form for a response. For details of the areas in your API definition from where you can edit a response, see the following topics:

- Defining responses for an operation
- Defining response components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Response name: A name for the response.
     If you are editing an operation response, this name is the HTTP status code.

     If you are editing a response component, this name defines a key that enables this response to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/responses/Name`

     To change the name, click Update, then click Save when done.

   - Description: A description of the response. You can use CommonMark syntax for rich text representation.
   - Headers: Headers define information that can be returned in the response.
     To define a new header for the response, click Add, then refer to Creating a header.

     To edit an existing header, click the header name, then refer to Editing a header.

   - Content: A content definition describes the content of the response.
     To define a new content definition for the response, click Add, then refer to Creating a content definition.

     To edit an existing content definition, click the content definition name, then refer to Editing a content definition.

   - Links: Links describe how various values returned by one operation can be used as input for other operations.
     To define a new link for the response, click Add, then refer to Creating a link.

     To edit an existing link, click the link name, then refer to Editing a link.

2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Creating a content definition

A content definition describes the content of an API request, response, or parameter. You can create content definitions in various places in your API definition.

Before you begin

Launch the content definition creation window. For details of the areas in your API definition from where you can create a content definition, see the following topics:

- Editing a header
- Editing a request body
- Defining the request body for an operation
- Editing a response

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Content Type: Select the content type; for example, application/json.
     Note: While the Content Type field in the editor provides a selection list of media types, by editing the OpenAPI source for the API directly you can specify a media type range; for example, `text/*`, `*/json`, or `*/*`. For requests or responses that match multiple types, only the most specific type is applicable;. for example, `text/plain` overrides `text/*`.
2. Click Create.
   The content definition details are displayed for further editing; see Editing a content definition.

Editing a content definition

A content definition describes the content of an API request, response, or parameter. You can edit content definitions that have been previously created in various places in your API definition.

Before you begin

Open the details form for a content definition. For details of the areas in your API definition from where you can edit a content definition, see the following topics:

- Editing a header
- Editing a request body
- Defining the request body for an operation
- Editing a response

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Content Type: Select the content type; for example, application/json.
     Note: While the Content Type field in the editor provides a selection list of media types, by editing the OpenAPI source for the API directly you can specify a media type range; for example, `text/*`, `*/json`, or `*/*`. For requests or responses that match multiple types, only the most specific type is applicable;. for example, `text/plain` overrides `text/*`.
   - Schema:
     To define a schema for the content definition, click Create, then refer to Creating a schema component.

If the content definition already has a schema defined, click View to edit the schema.

For full details on editing a schema, see Editing a schema component.
Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see Defining schema components. You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.

- Example Value: An example of a parameter value. Whatever you enter here is displayed as-is in the Developer Portal.
- Examples:
  To add an example of the content, click Add, then refer to Creating an example.

  To edit an existing example, click the example name, then refer to Editing an example.

- Encoding: An encoding definition describes how a specific schema property is encoded.
  To add an encoding definition, click Add, then refer to Creating an encoding definition.

  To edit an existing encoding definition, click the name, then refer to Editing an encoding definition.

2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Creating an encoding definition

An encoding definition defines a map between a specific schema property and its encoding information.

Before you begin

Launch the encoding definition creation window when configuring a content definition; see Editing a content definition.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: The name of the schema property.
   - Content Type: The Content-Type for encoding the property; for example, `application/xml; charset=utf-8`, `image/png, image/jpeg`.
   - Style: Describes how the property value will be serialized depending on the type of the property value. Select one of the following options:
     - form: Form style properties defined by RFC6570.
     - spaceDelimited: Space separated array values.
     - pipeDelimited: Pipe separated array values.
     - deepObject: Provides a simple way of rendering nested objects using form properties.
   - Explode: When selected, property values of type `array` or `object` generate separate properties for each value of the array or key-value pair of the map.
   - Allow Reserved: Determines whether the property value should allow reserved characters, as defined by RFC3986 (`:/?#[]@!$&'()*+,;=`) to be included without percent-encoding.
2. Click Create.
   The encoding definition details are displayed for further editing; see Editing an encoding definition.

Editing an encoding definition

An encoding definition defines a map between a specific schema property and its encoding information

Before you begin

Open the details form for an encoding definition when configuring a content definition; see Editing a content definition.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: The name of the schema property. To change the name, click Update, then click Save when done.
   - Content Type: The Content-Type for encoding the property; for example:
     - `application/xml; charset=utf-8`
     - `image/png, image/jpeg`.
   - Headers: A map allowing additional information to be provided as headers.
     To add a new header, click Add, then refer to Creating a header.

     To edit an existing header, click the header name, then refer to Editing a header.

   - Style: Describes how the property value will be serialized depending on the type of the property value. Select one of the following options:
     - form: Form style properties defined by RFC6570.
     - spaceDelimited: Space separated array values.
     - pipeDelimited: Pipe separated array values.
     - deepObject: Provides a simple way of rendering nested objects using form properties.
   - Explode: When selected, property values of type `array` or `object` generate separate properties for each value of the array or key-value pair of the map.
   - Allow Reserved: Determines whether the property value should allow reserved characters, as defined by RFC3986 (`:/?#[]@!$&'()*+,;=`) to be included without percent-encoding.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining parameter components

Parameter components define reusable parameters that can specify variable elements of a URL path, query parameters, headers, or a request body.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a parameter component, you must reference it from elsewhere in your API definition.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to `header`.
- A parameter can have various format styles, the style of a header is always `simple`.

For more information on headers, see Creating a header.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more parameter components defined, expand Parameters.
3. You can create a new parameter component, or edit any existing parameter component.
   - To create a parameter component, click the add icon ⊕ alongside Parameters, then refer to Creating a parameter.
   - To edit an existing parameter component, click the parameter component name in the navigation pane, then refer to Editing a parameter.

- **Creating a parameter**
  Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.
- **Editing a parameter**
  Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for Paths and Path operations in your API definition.

Creating a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can create parameters for Paths and Path operations in your API definition.

Before you begin

Launch the parameter creation window. For details of the areas in your API definition from where you can create a parameter, see the following topics:

- Defining parameters for a Path
- Defining parameters for an operation
- Defining parameter components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

For more information on headers, see Creating a header.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the parameter; you can either create a parameter from scratch, or you can reference a parameter that is defined in a parameter component.
   - To create the parameter from scratch, provide the following information on the Definition tab:
     - Name: Available if you are creating a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/parameters/`*`Name`*

     - Parameter name (required): The name of the parameter.
     - Located In (required): The location of the parameter. Select one of the following options:
       - query: Parameters that are appended to the URL. For example, ***`url_path`*`?myparam=myvalue`**.
       - header: Custom headers that are expected as part of the request.
       - path: The parameter value is part of the operation's URL, enclosed in {}.
       - cookie: Used to pass a specific cookie value to the API.
     - Description: A description of the parameter. You can use CommonMark syntax for rich text representation.
     - Style: Describes how the parameter value will be serialized depending on the type of the parameter value. Select one of the following options:
       - matrix: Path-style parameters defined by RFC6570.
       - label: Label style parameters defined by RFC6570.
       - simple: Simple style parameters defined by RFC6570.
       - form: Form style parameters defined by RFC6570.
       - spaceDelimited: Space separated array values.
       - pipeDelimited: Pipe separated array values.
       - deepObject: Provides a simple way of rendering nested objects using form parameters.
     - Select the following options as required:
       - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
       - Deprecated: Specifies that a parameter is deprecated and should be transitioned out of usage.
       - Allows Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
       - Explode: When selected, parameter values of type **`array`** or **`object`** generate separate parameters for each value of the array or key-value pair of the map.
       - Allow Reserved: Determines whether the parameter value should allow reserved characters, as defined by RFC3986 (**`:/?#[]@!$&'()*+,;=`**) to be included without percent-encoding. This property applies only if the Located In property is set to query.
   - To reference a parameter that is defined in a parameter component, provide the following information on the Reference tab:
     - Name: Available if you are creating a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/parameters/`*`Name`*

     - Parameter name (required, available if you are creating a path or operation parameter): The name of the parameter.
     - Reference from: Select a reference to a parameter component.
2. Click Create.
   The parameter details are displayed for further editing; see Editing a parameter.

Editing a parameter

Parameters define variable elements of a URL path, query parameters, headers, or a request body. You can edit parameters that have been previously created for Paths and Path operations in your API definition.

Before you begin

Open the details form for a parameter. For details of the areas in your API definition from where you can edit a parameter, see the following topics:

- Defining parameters for a Path
- Defining parameters for an operation
- Defining parameter components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A parameter is similar to a header, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

For more information on headers, see Creating a header.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: Available if you are editing a parameter component, this name defines a key that enables this parameter to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/parameters/`*`Name`*

     To change the name, click Update, then click Save when done.
   - Parameter name (required): The name of the parameter.
   - Located In (required): The location of the parameter. Select one of the following options:
     - query: Parameters that are appended to the URL. For example, ***`url_path`*`?myparam=myvalue`**.
     - header: Custom headers that are expected as part of the request.
     - path: The parameter value is part of the operation's URL, enclosed in {}.
     - cookie: Used to pass a specific cookie value to the API.
   - Description: A description of the parameter. You can use CommonMark syntax for rich text representation.
   - Style: Describes how the parameter value will be serialized depending on the type of the parameter value. Select one of the following options:
     - matrix: Path-style parameters defined by RFC6570.
     - label: Label style parameters defined by RFC6570.
     - simple: Simple style parameters defined by RFC6570.
     - form: Form style parameters defined by RFC6570.
     - spaceDelimited: Space separated array values.
     - pipeDelimited: Pipe separated array values.
     - deepObject: Provides a simple way of rendering nested objects using form parameters.
   - Select the following options as required:
     - Required: Determines whether this parameter is mandatory. If the Located In property is set to path, this option must be selected.
     - Deprecated: Specifies that a parameter is deprecated and should be transitioned out of usage.
     - Allows Empty Value: Allows sending a parameter with an empty value. This is valid only if the Located In property is set to query.
     - Explode: When selected, parameter values of type **`array`** or **`object`** generate separate parameters for each value of the array or key-value pair of the map.
     - Allow Reserved: Determines whether the parameter value should allow reserved characters, as defined by RFC3986 (**`:/?#[]@!$&'()*+,;=`**) to be included without percent-encoding. This property applies only if the Located In property is set to query.
   - Schema:
     To define a schema for the parameter, click Create, then refer to Creating a schema component.

     If the parameter already has a schema defined, click View to edit the schema.

     For full details on editing a schema, see Editing a schema component.
     Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see Defining schema components. You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.
   - Example Value: An example of a parameter value. Whatever you enter here is displayed as-is in the Developer Portal.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining example components

An example component is a reusable example that you can add to an API to make the specification of your API clearer.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of an example component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more example components defined, expand Examples.
3. You can create a new example component, or edit any existing example component.
   - To create an example component, click the add icon ⊕ alongside Examples, then refer to Creating an example.
   - To edit an existing example component, click the example component name in the navigation pane, then refer to Editing an example.

- **Creating an example**
  You add examples to an API to make the specification of your API clearer.
- **Editing an example**
  You add examples to an API to make the specification of your API clearer.

Creating an example

You add examples to an API to make the specification of your API clearer.

Before you begin

Launch the example creation window. For details of the areas in your API definition from where you can create an example, see the following topics:

- Editing a content definition
- Defining example components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the example; you can either create an example from scratch, or you can reference an example that is defined in an example component.
   - To create the example from scratch, provide the following information on the Definition tab:
     - Example Name: If you are creating an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/examples/Name`

     - Summary: A summary description of the example.
     - Description: A full description of the example. You can use CommonMark syntax for rich text representation.
     - External Value: A URL that points to a literal example. This provides the capability to reference examples that cannot easily be included in JSON or YAML documents. This could a PDF, for example.

- To reference an example that is defined in an example component, provide the following information on the Reference tab:
  - Example Name: If you are creating an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:

    `#/components/examples/Name`

  - Reference from: Select a reference to an example component.

2. Click Create.
   The example details are displayed for further editing; see Editing an example.

Editing an example

You add examples to an API to make the specification of your API clearer.

Before you begin

Open the details form for an example. For details of the areas in your API definition from where you can edit an example, see the following topics:

- Editing a content definition
- Defining example components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   -
     - Example Name: If you are editing an example component, this name defines a key that enables this example to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/examples/Name`

       To change the name, click Update, then click Save when done.
     - Summary: A summary description of the example.
     - Description: A full description of the example. You can use CommonMark syntax for rich text representation.
     - Example Value: A literal example.
     - External Value: A URL that points to a literal example. This provides the capability to reference examples that cannot easily be included in JSON or YAML documents. This could a PDF, for example.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining request body components

A request body component is a reusable definition of the structure of the body of an API request.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more request body components defined, expand Request Bodies.
3. You can create a new request body component, or edit any existing request body component.
   - To create a request body component, click the add icon ⊕ alongside Request Bodies, then refer to Creating a request body.
   - To edit an existing request body component, click the request body component name in the navigation pane, then refer to Editing a request body.

- **Creating a request body**
  A request body defines the structure of the body of an API request.
- **Editing a request body**
  A request body defines the structure of the body of an API request.

Creating a request body

A request body defines the structure of the body of an API request.

Before you begin

Launch the request body creation window; see Defining request body components.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the request body; you can either create a request body from scratch, or you can reference a request body that is defined in a request body component.
   - To create the request body from scratch, provide the following information on the Definition tab:
     - Name: A name for the request body.
       If you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/requestBodies/Name`

     - Description: A description of the request body. You can use CommonMark syntax for rich text representation.
     - Required: Determines if the request body is required in the request.
   - To reference a request body that is defined in a request body component, provide the following information on the Reference tab:
     - Name: A name for the request body.
       If you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/requestBodies/Name`

     - Reference from: Select a reference to a request body component.
2. Click Create.
   The request body details are displayed for further editing; see Editing a request body.

Editing a request body

A request body defines the structure of the body of an API request.

Before you begin

Open the details form for a request body; see Defining request body components.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: A name for the request body.
     Available if you are creating a request body component, this name defines a key that enables this request body to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/requestBodies/Name`

   - Description: A description of the request body. You can use CommonMark syntax for rich text representation.
   - Content: A content definition describes the content of the request body.
     To define a new content definition for the request body, click Add, then refer to Creating a content definition.

     To edit an existing content definition, click the content definition name, then refer to Editing a content definition.

   - Required: Determines if the request body is required in the request.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining header components

A header component defines a reusable custom header that can be sent in an API request or returned in an API response.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a header component, you must reference it from elsewhere in your API definition.

A header is similar to a parameter, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to `header`.
- A parameter can have various format styles, the style of a header is always `simple`.

For more information on parameters, see Creating a parameter.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more header components defined, expand Headers.
3. You can create a new header component, or edit any existing header component.
   - To create a header component, click the add icon alongside Headers, then refer to Creating a header.
   - To edit an existing header component, click the header component name in the navigation pane, then refer to Editing a header.

- **Creating a header**
  A header defines information that can be sent in an API request or returned in an API response. You can create headers in various places in your API definition.
- **Editing a header**
  A header defines information that can be sent in an API request or returned in an API response. You can edit headers that have been previously created in various places in your API definition.

Creating a header

A header defines information that can be sent in an API request or returned in an API response. You can create headers in various places in your API definition.

Before you begin

Launch the header creation window. For details of the areas in your API definition from where you can create a header, see the following topics:

- Editing a response
- Editing an encoding definition

- Defining header components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A header is similar to a parameter, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

For more information on parameters, see Creating a parameter.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the header; you can either create a header from scratch, or you can reference a header that is defined in a header component.
   - To create the header from scratch, provide the following information on the Definition tab:
     - Header name: If you are creating a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/headers/`*`Name`*

     - Description: A description of the header. You can use CommonMark syntax for rich text representation.
     - Required: Determines whether this header is mandatory.
     - Deprecated: Specifies that this header is deprecated and should be transitioned out of usage.
     - Allow Empty Value: The header can be sent with an empty value.
     - Explode: When selected, header values of type **`array`** or **`object`** generate separate headers for each value of the array or key-value pair of the map.
     - Allow Reserved: Determines whether the header value should allow reserved characters, as defined by RFC3986 (**`:/?#[]@!$&'()*+,;=`**) to be included without percent-encoding.
   - To reference a header that is defined in a header component, provide the following information on the Reference tab:.
     - Header name: If you are creating a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/headers/`*`Name`*

     - Reference from: Select a reference to a header component.
2. Click Create.
   The header details are displayed for further editing; see Editing a header.

Editing a header

A header defines information that can be sent in an API request or returned in an API response. You can edit headers that have been previously created in various places in your API definition.

Before you begin

Open the details form for a header. For details of the areas in your API definition from where you can edit a header, see the following topics:

- Editing a response
- Editing an encoding definition
- Defining header components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

A header is similar to a parameter, with the following differences:

- Parameters carry actual data and are available to end users, headers carry meta data associated with a request or response and are hidden from end users.
- A parameter can have various locations, the location of a header is always set to **`header`**.
- A parameter can have various format styles, the style of a header is always **`simple`**.

For more information on parameters, see Creating a parameter.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Header name: If you are editing a header component, this name defines a key that enables this header to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/headers/Name`

     To change the name, click Update, then click Save when done.
   - Description: A description of the header. You can use CommonMark syntax for rich text representation.
   - Required: Determines whether this header is mandatory.
   - Deprecated: Specifies that this header is deprecated and should be transitioned out of usage.
   - Allow Empty Value: The header can be sent with an empty value.
   - Explode: When selected, header values of type **array** or **object** generate separate headers for each value of the array or key-value pair of the map.
   - Allow Reserved: Determines whether the header value should allow reserved characters, as defined by RFC3986 (`:/?#[]@!$&'()*+,;=`) to be included without percent-encoding.
   - Schema:
     To define a schema for the header, click Create, then refer to Creating a schema component.

     If the header already has a schema defined, click View to edit the schema.

     For full details on editing a schema, see Editing a schema component.
     Note: You cannot edit the schema if it references a schema component, the schema configuration is inherited from the schema component; for details on configuring a schema component, see Defining schema components. You can, however, edit the referenced schema component but any changes will be reflected anywhere that the schema component is referenced.
   - Content: A content definition describes the content of the header.
     To define a new content definition for the header, click Add, then refer to Creating a content definition.

     To edit an existing content definition, click the content definition name, then refer to Editing a content definition.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Defining security scheme components

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

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create security definitions of the following types:

Type	Description
Basic authentication	Use a basic authentication security definition to specify a user registry or an authentication URL to be used to authenticate access to the API.
API key	Use an API key security definition to specify what application credentials are required to call an API.
OAuth2	Use an OAuth2 security definition to specify settings for OAuth token based authentication for your API.
HTTP Bearer	Use an HTTP Bearer definition to specify how to validate the Bearer token that is required to call an API.

- **Defining basic authentication security scheme components**
  A basic authentication security scheme component is used when an application that calls the API is required to authenticate through a user registry.
- **Defining API key security scheme components**
  An API key security component is used to specify the credentials that an application must provide to identify itself when calling the API operations.
- **Defining OAuth2 security scheme components**
  An OAuth2 security scheme component defines the settings for controlling access to the API operations through the OAuth authorization standard.
- **Defining an HTTP bearer security scheme**
  An HTTP bearer security scheme is used to generate access tokens that are exchanged between the server and the client when calling the API operations.

Defining basic authentication security scheme components

A basic authentication security scheme component is used when an application that calls the API is required to authenticate through a user registry.

Before you begin

For authenticating access to the operations of APIs that are enforced on the API Connect gateway, the following user registry types are supported:

- Authentication URL
- LDAP

The following user registry types are **not** supported:

- OpenID Connect (OIDC)
- Local User Registry (LUR)
- A custom user registry

Before you can create a basic authentication security definition in an API, the user registry must exist. To create a user registry, you can use either API Manager or Cloud Manager. When you create a registry in API Manager, it is visible only to your provider organization. When you create a registry in Cloud Manager, you can make it visible to multiple provider organizations.

To create a user registry with API Manager, see Working with user registries.

To create a user registry with Cloud Manager, see User registries overview.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you use basic authentication, you require API users to provide a valid user name and password to access selected operations. The application developer must also provide an HTTP authorization header in requests that are sent to operations that require basic authentication.

When you use an authentication URL, the user credentials that are provided in the authorization header are validated by the endpoint specified in the URL. If the user is authenticated, IBM API Connect expects an authentication URL to return an HTTP `200 OK` response status code. All other HTTP response status codes result in an authentication failure and access is denied.

For more information about using an LDAP user registry for authentication, see LDAP authentication.

For information about using an Authentication URL, see Authentication URL user registry.

To make use of a basic authentication security scheme component, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.
3. To create a basic authentication security scheme component, click the add icon ⊕ alongside Security Schemes. To edit an existing basic authentication security scheme component, click the component name in the navigation pane.
4. If you are creating a new basic authentication security scheme component, you can either create it from scratch or you can reference another basic authentication security scheme component.
   To create a basic authentication security scheme component from scratch, provide the following information on the Definition tab:
   - Security Scheme Name (required): A key that enables this basic authentication security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/`*`Security_Scheme_Name`*

   - Security Scheme Type (required): Select http.
   - Scheme (required): The name of the HTTP Authorization scheme to be used in the Authorization header as defined in RFC7235.
   - Authenticate using User Registry: The name of the user registry to be used for authentication. This is the value of the **`name`** field that was auto-generated when the user registry was created.
   - Bearer format: A hint to the client to identify how the bearer token is formatted. Bearer tokens are usually generated by an authorization server, so this information is primarily for documentation purposes.
   - Description: An optional description of the basic authentication security scheme. You can use CommonMark syntax for rich text representation.

   To reference another basic authentication security scheme component, provide the following information on the Reference tab:
   - Security Scheme Name (required): A key that enables this basic authentication security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/`*`Security_Scheme_Name`*

   - Reference from: Select a reference to another basic authentication security scheme component.
5. If you are creating a new basic authentication security scheme component, click Create.
   The security scheme component details are displayed for further editing.
6. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

Fore more information on LDAP and Authentication URL, see LDAP authentication and Authentication URL user registry.

Defining API key security scheme components

An API key security component is used to specify the credentials that an application must provide to identify itself when calling the API operations.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can require that, when calling an API operation, an application must provide either a client ID, or a client ID and client secret; you create an API key security scheme component to specify a credentials requirement. If you require that an application must provide both a client ID and client secret, you must create two API key security scheme components, one for each type of credentials.

To make use of an API key security scheme component, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.
3. To create an API key security scheme component, click the add icon alongside Security Schemes. To edit an API key existing security scheme component, click the component name in the navigation pane.
4. If you are creating a new API key security scheme component, you can either create it from scratch or you can reference another API key security scheme component.
   To create an API key security scheme component from scratch, provide the following information on the Definition tab:
   - Security Scheme Name: A key that enables this API key security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/Security_Scheme_Name`

   - Security Scheme Type: Select apiKey.
   - Name: The name of the header, query or cookie parameter to be used. For a key of type client secret you must include the string "secret", ignoring case, in the value you specify in the Name field; if the value does not contain the string "secret" then the key is assumed to be of type client ID.
     Note: The client ID and client secret headers that are specified in the request when the API is called are **not** added automatically to the message context. If you need these headers in the message context for subsequent processing, include a `set-variable` policy in your API assembly that adds the headers to the message content, taking the values specified in the request; you can do this in either of the following ways:
     - In the Assemble tab, add a Set Variable policy to your API assembly that defines the appropriate actions; for example:

       Table 1.

Action	Set	Type	Value
Set	message.headers.X-IBM-Client-Id	string	$(request.headers.X-IBM-Client-Id)
Set	message.headers.X-IBM-Client-Secret	string	$(request.headers.X-IBM-Client-Secret)

       For more information on configuring a Set Variable policy, see Set Variable.
     - In the Source tab, add a `set-variable` policy directly to the assembly section in your OpenAPI source; for example:

```
assembly:
  execute:
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: message.headers.X-IBM-Client-Id
            value: $(request.headers.X-IBM-Client-Id)
            type: string
          - set: message.headers.X-IBM-Client-Secret
            value: $(request.headers.X-IBM-Client-Secret)
            type: string
```

       For more information on defining a `set-variable` policy in your OpenAPI source, see set-variable.
     You should position the `set-variable` before any policy that needs to access the API key headers; for example, an invoke policy that calls a back-end service that is required to identify the application that made the initial request.
   - Key Type: Select client_id or client_secret.
     Important: The client_id is not confidential and functions as a "SecurityScheme" only in the meaning of the OpenAPI specification.
     If you use only the client_id (without the client_secret) as your API Key, be aware that the client_id is not considered a confidential value in the API Manager UI; for example, it might be displayed to other authenticated members of the provider organization. The client_id is treated as a public identifier for apps and cannot be kept private.

     If authentication of the identity of the app calling an API is needed, then the API Key should use both a client_id to identify the app, plus a client_secret to authenticate it. The client_secret is designed to be known only to the app and the authorization server, and it should be protected. IBM API Connect protects

the client_secret: it is only available when first generated, it can be stored as a one-way hash to prevent retrieval or leaks, and it can be changed and reset without changing the corresponding client_id. When an API Key is not marked as a client_secret, it will not be protected in the API Manager UI.

- Located In: Select one of the following options, to specify how the credentials are sent:
  - header: The credentials are sent in the request header.
  - query: The credentials are sent as query parameters. This method is less secure because the client secret could be exposed in a log file.

  Note: If you migrated from API Connect V5, API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. For information about the differences, see Client ID behavior in API key security definitions.
- Description: An optional description of the API key security scheme. You can use CommonMark syntax for rich text representation.

5. To reference another API key security scheme component, provide the following information on the Reference tab:
   - Security Scheme Name: A key that enables this API key security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/Security_Scheme_Name`

   - Reference from: Select a reference to another API key security scheme component.
6. If you are creating a new API key security scheme component, click Create.
   The security scheme component details are displayed for further editing.
7. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

- **Client ID behavior in API key security definitions**
  How the Client ID behaves in API key security definitions in IBM API Connect 10 compared to Version 5.

Client ID behavior in API key security definitions

How the Client ID behaves in API key security definitions in IBM® API Connect 10 compared to Version 5.

API key security definitions that contain multiple Client ID references, behave differently in Version 10 than in Version 5. Refer to the following table to understand the differences in Client ID behavior for different use cases.
Note: Although a client_id in the request body is supported, this data is not used during security requirement processing.

Table 1. Client ID behavior in V5 and V10

Use case	Description of security requirement	V5 behavior	V10 behavior
1	Allows client_id in the query, and the request contains multiple client_ids in the query.	Uses the first client ID in the request.	401 invalid client ID or secret.
2	Allows client_id in the header, and the request contains multiple client_ids in the header.	401 invalid client ID or secret.	401 invalid client ID or secret.
3	Allows client_id in the header, and the request contains multiple client_ids in the query and the header (1 in the query, 1 in the header).	401 client ID in wrong location.	Client ID in query is ignored.
4	Allows client_id in the query, and the request contains multiple client_ids in the query and the header.	Client ID in header is ignored.	Client ID in header is ignored.
5	Allows client_id in the query or header, and both the query and header contain multiple client_ids.	Header returns 401 client ID in wrong location. Query uses the first client ID in the request.	401 invalid client ID or secret.
6	Allows client_id in the header, and the request contains a valid client_id in the query, and an invalid client_id in the header.	401 client ID in wrong location.	401 invalid client ID or secret.
7	Allows client_id in the query, and the request contains a valid client_id in the header, and an invalid client_id in the query.	401 invalid client ID or secret.	401 invalid client ID or secret.
8	Allows client_id in the query or header, and both the query and header contain valid client_ids.	200 OK.	403 multiple client IDs.
9	Allows client_id in the query or header, and both the query and header contain invalid client_ids.	401 client ID in wrong location.	401 invalid client ID or secret.
10	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the query is valid.	200 OK.	Uses the valid one 200 OK.
11	Allows client_id in the query or header, and both the query and header contain a client_id, and the one in the header is valid.	401 client ID in wrong location.	Uses the valid one 200 OK.
12	Allows client_id in the query or header, and both the query and client_id is missing in the request (no security requirement, but client security is defined in the assembly).	401 invalid client ID or secret.	• When Return V5 Responses compatibility option is off: 401 client ID is missing. • When Return V5 Responses compatibility option is on: 401 invalid client ID or secret. For information about setting compatibility options, see Configuring v5 compatibility options.

Defining OAuth2 security scheme components

An OAuth2 security scheme component defines the settings for controlling access to the API operations through the OAuth authorization standard.

Before you begin

Before you can create an OAuth2 security scheme component, you must:

1. Create an OAuth provider.
   - To use Cloud Manager, see Configuring a native OAuth provider or Configuring a third-party OAuth provider.
   - To use API Manager, see Configuring a native OAuth provider or Configuring a third-party OAuth provider.
2. Add the OAuth provider to a catalog. If you have not created any catalogs, use the Sandbox Catalog. See the OAuth instructions step in Creating and configuring Catalogs.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of an OAuth2 security scheme component, you must reference it from elsewhere in your API definition. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more security scheme components defined, expand Security Schemes.
3. To create an OAuth2 security scheme component, click the add icon ⊕ alongside Security Schemes. To edit an existing OAuth2 security scheme component, click the component name in the navigation pane.
4. If you are creating a new OAuth2 security scheme component, you can either create it from scratch or you can reference another OAuth2 security scheme component.
   To create an OAuth2 security scheme component from scratch, provide the following information on the Definition tab:
   - Security Scheme Name: A key that enables this OAuth2 security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/Security_Scheme_Name`

   - Security Definition Type: Select oauth2.
   - Flow: Select one or more of the following options:
     - Implicit
     - Resource Owner - Password
     - Application
     - Access Code

     Note: The flow options that are available for selection depend on which options are supported by the selected OAuth provider.
     If you selected Resource owner - JWT as the supported grant type when you created the OAuth provider, you must select Resource owner - Password as the Flow for this security scheme.
   - OAuth Provider: The name of the OAuth provider to be used. This is the value of the **name** field that was auto-generated when the OAuth provider was created.
   - Description: An optional description of the OAuth2 security scheme. You can use CommonMark syntax for rich text representation.
   - Optionally select your OIDC provider from the list to add the provider's URL to the OpenId Connect URL field on the Resources > OAuth Providers > OpenID Connect page, which enables a client to send a request for information about the provider's publicly visible configuration.

   To reference another OAuth2 security scheme component, provide the following information on the Reference tab:
   - Security Scheme Name: A key that enables this OAuth2 security scheme component to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/securitySchemes/Security_Scheme_Name`

   - Reference from: Select a reference to another OAuth2 security scheme component.
5. If you are editing an existing OAuth2 security scheme component, provide the following information:
   - The URL endpoints (the applicable endpoints depend on the flow type):
     - Authorization URL
     - Refresh URL
     - Token URL

     Note: The URL endpoint values supplied here are maintained only for informative purposes, no validation or other action is applied to them by API Connect. The OAuth security that is applied depends on the specified OAuth provider.
   - Scopes:
     - To add a new scope, click Add. To edit an existing scope, click the options icon ⋮ alongside the required scope, then click Edit.
     - Provide the following information:
       - Scope: The scope identifier.
       - Description: An optional description of the scope. You can use CommonMark syntax for rich text representation.
6. If you are creating a new OAuth2 security scheme component, click Create.
   The security scheme component details are displayed for further editing.
7. Click Save when done.

What to do next

Apply the security scheme to an API or operation. For more information, see Enforcing security requirements on an API and Enforcing security requirements on an operation.

Defining an HTTP bearer security scheme

An HTTP bearer security scheme is used to generate access tokens that are exchanged between the server and the client when calling the API operations.

About this task

For information on the use of bearer tokens with OpenAPI 3 APIs, see Bearer Authentication in the OpenAPI 3 specification.

If an HTTP bearer token is found in the request, its value is stored within the context as **`api->security->bearer_token`**. If an external URL validation is invoked, any response with a status code of 200 is stored within the context as **`api->security->bearer_validation_response`**.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components > Security Schemes.
3. Click Add.
4. In the Add Object dialog box, provide the following information:
   - Security Scheme Name (Key) - Provide a descriptive name for the new scheme.
   - Security Scheme Type - Select http.
   - Scheme - Select Bearer.
   - Bearer Format - Select JWT.
   - Validation Method - Select a method. If you select external-url, provide the following additional information:
     - Validation Endpoint - Provide the URL of the server used for validation. To ensure a secure connect, the URL should use the **`https`** protocol.
     - TLS Profile name - If the validation endpoint uses the **`https`** protocol, select the name of the TLS client profile to use for a secure connection.
   - Description - Provide a description of the JWT scheme.
5. Click Create to create the new scheme.
6. Click Save in the page header.

Defining link components

A link component is a reusable link definition that can be used to describe how various values returned by one operation can be used as input for other operations. In this way, links provide a known relationship and traversal mechanism between the operations.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To make use of a link component, you must reference it from elsewhere in your API definition.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Expand Components then, if there are already one or more link components defined, expand Links.
3. You can create a new link component, or edit any existing link component.
   - To create a link component, click the add icon ⊕ alongside Links, then refer to Creating a link.
   - To edit an existing link component, click the link component name in the navigation pane, then refer to Editing a link.

- **Creating a link**
  Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.
- **Editing a link**
  Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

Creating a link

Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

Before you begin

Launch the link creation window. For details of the areas in your API definition from where you can create a link, see the following topics:

- Editing a response
- Defining link components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Create the link; you can either create a link from scratch, or you can reference a link that is defined in a link component.
   - To create the link from scratch, provide the following information on the Definition tab:
     - Name: If you are creating a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/links/Name`

     - Operation ID: The ID of an existing operation; this field is mutually exclusive of the Operation Ref field. For details on defining operations, see Defining operations for a Path.
     - Operation Ref: A relative or absolute URI reference to an existing operation; this field is mutually exclusive of the Operation ID field. For details on defining operations, see Defining operations for a Path.
     - Description: A description of the link. You can use CommonMark syntax for rich text representation.
   - To reference a link that is defined in a link component, provide the following information on the Reference tab:
     - Name: If you are creating a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:

       `#/components/links/Name`

     - Reference from: Select a reference to a link component.
2. Click Create.
   The link details are displayed for further editing; see Editing a link.

Editing a link

Links describe how various values returned by one operation can be used as input for other operations, providing a known relationship and traversal mechanism between the operations.

Before you begin

Open the details form for a link. For details of the areas in your API definition from where you can edit a link, see the following topics:

- Editing a response
- Defining link components

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.

- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Provide the following information:
   - Name: If you are editing a link component, this name defines a key that enables this link to be referenced from elsewhere in the API definition; the reference has the following format:

     `#/components/links/Name`

     To change the name, click Update, then click Save when done.
   - Operation ID: The ID of an existing operation; this field is mutually exclusive of the Operation Ref field. For details on defining operations, see Defining operations for a Path.
   - Operation Ref: A relative or absolute URI reference to an existing operation; this field is mutually exclusive of the Operation ID field. For details on defining operations, see Defining operations for a Path.
   - Parameters: A map representing parameters to pass to an operation as specified with Operation ID or identified via Operation Ref. The key is the parameter name to be used, whereas the value can be a constant or an expression to be evaluated and passed to the linked operation.
   - Request Body: The request body to use when calling the target operation.
   - Description: A description of the link. You can use CommonMark syntax for rich text representation.
   - Server: A server to be used by the target operation. Provide the following information:
     - Server URL The specified URL is used to determine the full URL endpoint for the calling the API, according to any vanity endpoint configuration in the Catalog in which the API is published. For full details, see Configuring vanity endpoints for a Catalog.
     - Server Description: An optional description of the host designated by the URL. You can use CommonMark syntax for rich text representation.
     - Server Variables A server variable defines a map between a variable name and its value. The value is used for substitution in the server's URL template.
       - To add a new server variable, click Add. To edit an existing server variable, click the variable name.
       - Provide the following information:
         - Server Variable Name (required).
         - Default value (required).
         - An optional rich text description. You can use CommonMark syntax for rich text representation.
         - One or more Enum Value entries (available when editing an existing server variable). Enum values specify an enumeration of string values to be used if the substitution options are from a limited set. To add an new Enum value, click Add, enter the value, then click Create. To edit an existing Enum value, click the Enum value.
       - If you are creating a new server variable, click Create. The server variable details are displayed for further editing.
2. Click Save when done.

What to do next

If required, use the breadcrumb trail to navigate to another location in the hierarchy of the object you are working on.

Specifying gateway and portal settings

Define general configuration settings for your API.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, then click Gateway and portal settings.
3. Provide the following information:
   - Phase: The phase of the lifecycle that your API is in; select one of the following options:
     - Realized (default) - The API is in the implementation phase.
     - Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
     - Specified - The API has been fully designed and passed an internal milestone but has not yet been implemented.
   - Testable: Select this option to allow the API's operations to be tested using the test tool in the Developer Portal.
     Note: For the test tool to work, an API must be included in a Plan in a Product that is staged in a Catalog.
   - CORS: Enable and configure cross-origin resource sharing (CORS) support for your API. For details, see Enabling CORS support for an API.
   - Target Services: Define web services that you want to use in your API definition. For details, see Defining target services for an API.

- Enforced: Select this option to enforce the API by using the DataPower API Gateway. Clear this option if you are managing the API on a gateway other than the DataPower API Gateway. Although not published to a gateway by API Connect, an unenforced API is still available in the Developer Portal for subscription by application developers.
- Properties: Add properties that can be referenced in your API definition. For details, see Setting API properties.
- Catalog Properties: Define property values that are specific to a particular Catalog. For details, see Defining Catalog specific property values.
- Activity Log: Configure your logging preferences for the API activity that is stored in analytics. For details, see Configuring activity logging.
- Exxtensions: Extensions added to the API to extend the OpenAPI specification. For details, see Adding an OpenAPI extension to an API.

Defining target services for an API

A target service defines a web service that you want to use in your API definition.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more target services defined, expand Target Services.
3. To define a new API target service, complete the following steps:
   a. Click the add icon ⊕ alongside Target Services in the navigation pane.
   b. Upload the service information from a stand-alone .wsdl file, or a .zip file that contains a WSDL file and its dependent documents, by either dragging and dropping your file, or browsing and selecting the file that you want to use.
   If you upload a .zip file, you can include in the .zip file an options file to specify additional directives. For details, see Using an options file when importing a WSDL service.
4. Select the service that you want to add.
5. Click Save to save your changes.
6. To view the source for an existing target service, click the target service name in the navigation pane.

Results

The target services that you add are now available in the palette on the Policies view for you to add to your API assembly flow; for more information, see Including elements in your API assembly.

Setting API properties

In addition to the pre-supplied API properties that you can use to control the behavior of API Connect policies, you can define your own API properties. The properties that you define can be referenced in your API definitions.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API properties include property name, value, and, optionally a specific Catalog to which a property value applies. For a list of pre-supplied API properties relating to various policies, see API properties.
Note: Once defined, an API property is read only.
For information on how to reference a property in an API definition, see Variable references in API Connect.

It is also possible to define properties that are specific to a Catalog and can be referenced by any of the APIs in that Catalog; for more information, see Creating and configuring Catalogs. Note that if you define a Catalog property of the same name as an API property, the API property takes precedence over the Catalog property.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more API properties defined, expand Properties.
3. Configure an API property.
   - To define a new API property, complete the following steps:
     - Click the add icon alongside Properties in the navigation pane.
     - Provide the following information:
       - Property Name: Enter a name for the property; this name is used to reference the property. The following character set is supported for the Name of an API property: `[A-Za-z0-9_-]+`. Spaces are allowed.
       - Value: A default value for the property. Leave blank if the property is to have a null vale by default.
       - Description: An optional description of the property. You can use CommonMark syntax for rich text representation.
       - Select Encoded if you want to hide the property values, or protect user passwords from casual observance.
         Note: If you encode a property value, it is saved in Base64 encoded form; it is **not** encrypted. If you subsequently clear the Encoded check box, the original property value is restored in its unencoded form.
     - Click Create.
   - To modify an existing API property, click the property name in the navigation pane. You can then change any of the configuration settings.
4. Click Save to save your changes.

What to do next

Optionally, define a value for the property that is specific to a particular Catalog; see Defining Catalog specific property values.

- **Defining Catalog specific property values**
  For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

Defining Catalog specific property values

For any API property, you can define a value that is specific to a particular Catalog. For any API published to that Catalog, the API property assumes the Catalog specific value, overriding any default property value.

Before you begin

You must have previously created the API property that you now want to define a Catalog specific value for; see Setting API properties.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Tip: If you add or change an API property on an API that is already staged or published, you must re-stage or re-publish the Product that contains the updated API for the change to take effect.

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings then, if there are already one or more Catalog specific property values defined, expand Catalog Properties.
3. Configure a Catalog specific property value.
   - To define a new Catalog specific property value, complete the following steps:
     - Click the add icon alongside Catalog Properties in the navigation pane.
     - Type the Catalog Name exactly as it displays in the **name** field for the selected Catalog.
       Tip: For the best result, manually type the catalog's name instead of selecting its title. Using the catalog name is safer than using its title, because the title might not be unique within the provider organization and might be modified later (which breaks the association between the property and the catalog). The catalog's name is always unique within the provider organization and cannot be changed.
     - Click Create. The Catalog properties details for the specified catalog are displayed for further editing.
     - Alongside Property Overrides, click Add.
     - Provide the following information:
       - Property Name: Enter the name of the API property that you want to define a Catalog specific value for. The name **must** match the name of a previously defined API property; see Setting API properties.
       - Property Value: The Catalog specific property value. Leave blank if the property is to have a null value by default.
     - Click Create.
   - To modify an existing Catalog specific property value, click the Catalog name in the navigation pane. You can then change the Catalog name, and add, modify, or delete property override settings.
4. Click Save to save your changes.

Configuring application authentication for an API

Application authentication settings allow you to protect your API with a certificate, for example, by using TLS mutual authentication. You can select whether a client certificate is sent as a TLS client certificate or in an HTTP header.

About this task

Note: This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Enable application authentication, as follows:
   Note: To identify the application, you must enable a security requirement of type **`clientID`**.
   a. Select the Gateway tab, expand Gateway and portal settings, then click Application Authentication.
   b. Select Certificate.
3. Specify how the client certificate is sent to the gateway, as follows:
   a. In the navigation pane, click Application Authentication Source.
      Any existing application authentication source definitions are listed.
   b. Alongside Application Authentication Source, click Add.
   c. Select the required option, header or tls-cert.
      You can separately add one or both of the options.
      When the API is called, an X509 client certificate must be supplied, either in the specified HTTP header, or as a TLS client certificate from TLS mutual authentication. For any Developer Portal application that calls the API, the certificate must be entered in the Developer Portal user interface; for details, see Registering an application.

      The Gateway service to which the API is published can be configured to use TLS mutual authentication to secure API calls made to that Gateway service; for details, see Configuring the initial Gateway service. If you select this option, continue to sub-step d.

      If you are using a load balancer, you must configure the load balancer to use the specified HTTP header to relay the appropriate client certificate to the Gateway service after the load balancer terminates the TLS communication.

   d. If you select the tls-cert option, enable mTLS at the Gateway level.
      This can be done by enabling "Mutual authentication: Required" in the TLS Server Profile configured in the Gateway Service in the Cloud Manager Topology (requires cloud administrator permissions). For information, see Registering a gateway service.

      You can create a Truststore (with the root or intermediate CA certificate) for use with the TLS Server Profile.

      The following example cURL command uses a client-cert during SSL:

      ```
      **curl -X GET 'https://GW-Server:9443/test/sandbox/mtls-testing/callmtls-call' -H 'X-IBM-Client-Id:
      1872xxxxxxxxxxxx' --cert my-leaf-cert.pem --key my-leaf-cert.key -k -v**
      ```

      Note: This feature is intended for the scenario where different applications will be registered with different end certificates in the Developer Portal, which will have the same intermediate or root CA issuer. The certificate will pass Gateway-level mTLS if it matches the issuer available in the Truststore, but will only allow access the to API that matches the correct end certificate.
4. Click Create.
5. Click Save to save your changes.

Configuring activity logging

You can configure your logging preferences for the API activity that is stored in analytics, overriding the default activity logging behavior.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

An API event record exists for each API execution event in the Gateway server. By default, the content type for a successful API execution is activity, and payload for an API call the results in an error code. When you compose your API definition, you can change the type of content to log in these API event records. During API execution, the activity data is stored in the **`log`** context variable, which populates the API event record on completion of the API execution; for more information, see API activity logging context variables.

Note:
Activity logging that calls for logging of analytics data upon success doesn't apply for the OAuth provider. The OAuth provider logs analytics data for failure cases, but doesn't log successful cases.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon . To return to the design form, click the Form icon .

Procedure

1. Open the API for editing, as described in Editing an OpenAPI 3.0 API definition.
2. Select the Gateway tab, expand Gateway and portal settings, then click Activity Log.
3. Select Enabled, then select options for the Success Content and Error Content fields as follows:

Field label	Description
Success Content	Defines the type of content to be logged when the operation is successful. • none: API events are not logged. • activity: Logs the API resource URI. • header: Logs the API resource URI and HTTP headers. • payload: Logs API resource URI, HTTP headers, and payload; all information, including the payload that is received in a request or returned in a response, is recorded. The default value is activity.
Error Content	Indicates what content to log when an error occurs. • none: API events are not logged. • activity: Logs the API resource URI. • header: Logs the API resource URI and HTTP headers. • payload: Logs API resource URI, HTTP headers, and payload; all information, including the payload that is received in a request or returned in a response, is recorded. The default value is payload.

4. Click Save to save your changes.

Including elements in your API assembly

An assembly is formed of elements that are applied to calls to and responses from operations in your API. Elements can be either policies or logic constructs.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

In the Policies view of IBM API Connect, you can add and configure elements in your assembly. You can also directly add elements to the OpenAPI definition of your API.

- **The assembly editor**
  The API Designer of IBM API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.
- **Adding elements to your assembly**
  Create your API assembly by using the Assemble view in the API Designer. You can access the Assemble view by clicking the Gateway tab in the API editor.
- **Handling errors in the assembly**
  Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

The assembly editor

The API Designer of IBM® API Connect features a graphical editor that you can use to create assemblies. With assemblies, you can readily tailor your APIs to include elements such as activity logging and redaction of specific fields.

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM API Connect.

You can access the Assemble view by clicking the Gateway tab in the API editor, and then click Policies. This tab includes the following options:

- Palette - lists the available elements in the assembly flow.
- Canvas - used to arrange and visualize the elements in the assembly.
- Property sheet - used to configure the elements in the assembly.

The palette

The palette is a list of different elements that you can include in your assembly. To view the palette, click the plus icon on the canvas. You can use the Rotate icon to rotate the palette as you prefer. To maximize and minimize the palette view, click the Maximize and Minimize icons.

You can release the palette by clicking the Close icon .

The canvas

You can use the canvas to create a graphical representation of the assembly flow. To add an element to the assembly flow from the palette, click the plus icon ⊕ at the appropriate location on the canvas and select the required element from the available list. To delete an element from the canvas, hover over the element and click the Delete icon 🗑.

Note: You must click Save in the Assemble view to save your assembly updates, and to make them appear in the Source view.
API calls are made at the initial, unfilled, circle, and returned at the final, filled, circle. You can insert elements between the two circles to modify the data received from the call, or returned by the response.

You can use the Show catches toggle to show and hide error catches in the palette. A catch is a section of the assembly that is applied when an API call results in the corresponding HTTP status code being returned. Click the catch section to open the property sheet for all your catches.

You can zoom the view of your canvas in and out by clicking the + and - icons. To fit the canvas to the screen size, click the Fit to screen icon ⛶.

The property sheet

When you select an element that is in the assembly by clicking it, details about the element are displayed in the property sheet. In this pane, you can configure the element's properties. The options available to you in the property sheet are specific to the type of element you are working with. For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down menu.

You can release the property sheet by clicking the Close icon ✕.

The behavior of an assembly

The assembly runs policies in order and acts on different contexts of the API call. When an API call is made, security and rate limits are enforced before the assembly is executed. During the assembly, the flow can be branched, thrown, and caught according to the policies available. The message context flows through the assembly, being used and altered by various policies. In addition to the message, other contexts can be accessed and created.

Security and rate limiting
: Before the assembly is executed, security and then rate limits are enforced.
First, security definitions and CORS access control are used to authenticate an API call. Any API Key security definitions are used to identify applications that have subscriptions to a Product containing the API. If a security definition does not allow access, the API call is rejected.

If an application is identified by its client ID or client secret, a rate limit can be enforced based on the Plan or operation called.

The assembly execution sequence
: The assembly is executed in order from the initial, filled, circle to the final, unfilled, circle. However, there is room for branching, when **`if`** and **`operation-switch`** logic constructs are used, or for the remaining assembly to be ignored when a **`throw`** policy is executed.
The message is the context that is acted upon by any policy that isn't otherwise configured. At the beginning of the API call, the message is empty and, at the end of the API call, the message is used as the response.

The request context contains the information that is sent by the API caller and varies with the type of operation called and the configuration of that operation. For example, a GET operation can never have a populated **`request.body`**, and you can configure an operation to have **`request.parameters`** (query parameters).
The first policy in an assembly acts on the request and produces the first instance of the message. If there are no policies, the request is returned to the caller.

Managing contexts
: Because the message can be overwritten, it is useful to create and reference new contexts where possible so that they are saved and reused during the API call.

- Use the **`map`** policy to overwrite the message context when you need to execute a policy that only acts on the message.
- Use the request context when you want to use the original request made to the API.
- Use the **`map`**, **`invoke`**, and **`proxy`** policies to create new contexts when you want to save your message.
  Note: When you create a new context, unless you are also mapping to the message, the message is overwritten with an empty object.

For example, **`invoke`** is the first policy in the assembly and its response overwrites the request as the message. The message is then acted upon by a **`validate`** policy, and a **`map`** policy then saves the message as a new context, ready for a second **`invoke`** policy to overwrite the message without losing the first **`invoke`** policy's output.

You can also access contexts outside of the message or your custom contexts, but these contexts cannot be written to. For a list of contexts, see API Connect context variables.

Branches and catches
: Using logic constructs, such as **`operation-switch`** or **`if`**, you can execute different sections of the assembly when certain conditions are fulfilled. When the assembly branches, the subsection of the assembly contained by the construct is executed in the same manner as a complete assembly. However, contexts are shared with the complete assembly.
When a catch is triggered, either by an error occurring during the execution of a policy, or because a **`throw`** policy is encountered, the rest of the assembly flow is ignored. All contexts are shared by the catch being executed and when the end of the catch is reached, the API call is completed. There is no way to return from a catch to the rest of the assembly.

Related concepts

- Variable references in API Connect

Related tasks

- Adding elements to your assembly

Related reference

- API Connect context variables

Adding elements to your assembly

Create your API assembly by using the Assemble view in the API Designer. You can access the Assemble view by clicking the Gateway tab in the API editor.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.
- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can use the assembly tool in the UI to create assemblies that are used to manipulate requests made to or responses made by any of your API's operations.

At any time, you can switch directly to the underlying OpenAPI YAML source by clicking the Source icon </>. To return to the design form, click the Form icon .

If you want to use the Source view to edit the source of your API definition, see execute OpenAPI extension for more information about how to configure the OpenAPI YAML.

For more information about the use of the assembly tool, see The assembly editor.

Procedure

To add elements to your assembly using the assembly tool, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Use the Policies view to add an element.
   Note: You must specify a version for the policy. To prevent errors due to incompatible versions of a policy between the API and the gateway, API Connect will only publish an API to the gateway if the policy version in the API exactly matches the version that is present on the gateway.
   a. Select the Gateway tab, then click Policies in the navigation pane.
   b. Click the plus icon ⊕ at the appropriate location on the canvas to view the palette that lists the available elements.
   c. Select the element that you want to add from the palette to the canvas by clicking it.
      Note:
      - Elements are applied in order from the initial, unfilled, circle to the final, filled, circle.
      - Unless an Operation Switch element is used, the whole assembly applies to every operation in the API.
   d. Add and edit properties of the element by clicking the element and using the property sheet.
      For some elements, you can add and remove properties by clicking Object Properties and selecting the property from the drop-down list. For information about the properties of policy elements, see API policies and logic constructs. For information about the properties of logic constructs, see Logic Constructs.
   e. Specify a version for the policy.
      To add a version, click the Source icon </>, and complete the `version` section of the policy YAML. For example:

      ```
      execute:
        - invoke:
            version: 2.2.0
            title: invoke
      ...
      ```

      You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions. For more information about how to configure the policy OpenAPI YAML, see execute OpenAPI extension.
4. Optional: Repeat Steps 3.b to 3.e for any additional elements that you want to add.
5. Click Save to save your changes.

Results

You have added one or more elements to your assembly.

Handling errors in the assembly

Use the catch section of the assembly to describe the handling of errors thrown during the assembly execution.

About this task

Note:

- This task relates to configuring an OpenAPI 3.0 API definition. For details on how to configure an OpenAPI 2.0 API definition, see Editing an OpenAPI 2.0 API definition.

- OpenAPI 3.0 APIs are supported only with the DataPower® API Gateway, not with the DataPower Gateway (v5 compatible).
- For details of current OpenAPI 3.0 support limitations, see OpenAPI 3.0 support in IBM® API Connect.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The catch section of the assembly is used to implement an assembly in the instance that an error is thrown during the assembly execution. For example, the assembly could contain a **throw** element, the API caller could fail to authenticate, or a policy could fail to execute correctly. Each error can be handled with a different catch and each catch can handle multiple status errors.

Procedure

To create a catch and include elements in it, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to apply a catch to.
3. Select the Gateway tab, then click Policies in the navigation pane.
4. Set the Show catches toggle in the menu bar of the canvas to the Show position.
5. Click Catches in the canvas, or a Catch icon if one is displayed.
   The property sheet for the API's catches opens.
6. To add a default catch that is executed when an otherwise uncaught error is thrown, click Add default catch.
   Note: If you have a default catch before another catch in precedence, the default catch will activate even when the other catch's error is thrown.
7. To add a new catch, click Add catch.
8. To specify which errors the catch applies to, use the Select errors field to select the appropriate error.
9. Optional: To remove an error case from a catch, clear the corresponding checkbox from the Select errors drop-down. To clear all the error cases from a catch, click the cross corresponding to the total number of error cases in the Select errors field.
10. Optional: To change the precedence of your catches, use the Move up or Move down icons.
    If an error case is handled by multiple catches, the catch at the beginning of the list is applied.
11. To add an element to a catch from the palette, click the plus icon at the appropriate location in the assembly flow. Select an element that you want to apply to the catch by clicking it.
12. Click Save to save your changes.

What to do next

If you added a catch for ConnectionError, SOAPError, or OperationError, you must add the same error to the Stop on error setting for the **Invoke** policy in your assembly. Otherwise, if the error occurs during the execution of the **Invoke** policy, it is not caught, the policy execution is allowed to complete, and the assembly flow continues. For details on configuring an **Invoke** policy, see Invoke.

For details of all the errors that can be returned by the assembly and are available to the catch function, see Error cases supported by assembly catches.

Validating the OpenAPI YAML source

How to validate the OpenAPI YAML source against swagger specifications when you're editing an API.

About this task

When you're configuring an API in the API editor, you can validate the OpenAPI YAML source by clicking Validate > With specifications. If any swagger parser validation errors are found, an error icon is displayed that you can then click to see the individual errors, and where they occur in the YAML. You can then fix the errors, save the updates, and continue to validate the OpenAPI YAML throughout the API configuration process.
Note:

- OpenAPI validation does not show gateway or policy errors that will stop your API from being staged or published. This type of validation occurs only when you try to stage or publish your API. For more information, see Staging an API or Publishing an API.
- You can also validate your API document by using API governance rulesets. For more information, see Validating an API document by using API governance.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Procedure

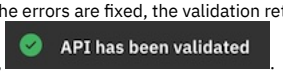
To validate the OpenAPI YAML source, complete the following steps.

1. Open the required API for editing, as described in Editing an OpenAPI 2.0 API definition or Editing an OpenAPI 3.0 API definition.
2. Click Validate > With specifications from the menu bar in the header.

3. If any swagger parser validation errors are found, an error icon is displayed on the header menu bar, with the number of OpenAPI errors found displayed next to the icon.

   For example, 1 .
4. You can then click the error icon to see a detailed list of the errors, and their location in the YAML. If you're in Form view, you can click an individual error to navigate directly to the relevant API section in the form.
5. After you fix the error, click Save to save your updates, and then click Validate again.

6. When all of the errors are fixed, the validation returns a success message.

   For example, 

.

Related concepts

- Testing an API

Related tasks

- Activating an API
- Staging an API

Validating an API document by using API governance

How to use API governance rulesets to validate and enforce organizational governance policies and best practices. You can validate APIs from the API Manager UI, the API Designer UI, or the toolkit CLI.

Before you begin

Before you can use API governance rulesets to validate your API documents, the API governance optional add-on must be enabled on your management subsystem by your system administrator. See Enabling API governance on Kubernetes, and Enabling API governance on VMware for more information. If API governance is enabled in your deployment, the API governance resource is displayed on the Resources page in the API Manager.

The API governance service is available to all user roles.

Tip: Before using the API governance service to validate your API document, you should run Validate > With specifications to validate the OpenAPI YAML source. For more information, see Validating the OpenAPI YAML source.

About this task

API governance is an IBM® API Connect service that can be used to validate and enforce organizational governance policies and best practices to your API development process. API governance contains the following types of rulesets:

- Provider organization rulesets - these are custom rulesets that contain the rules that are created in, and are specific to, your provider organization.
- Global rulesets - these are preconfigured IBM and Spectral rulesets that contain the rules that are shared with your provider organization, and cannot be edited. Note that the Spectral ruleset names are prefixed by spectral-, and that their version matches the version of that ruleset that's available in Spectral.

You can create your own provider organization rulesets to validate your Swagger, OpenAPI, and AsyncAPI documents against, or use the global rulesets that are provided by default. For more information about configuring rulesets, see Configuring API governance in the API Manager.

API governance in IBM API Connect is based on the open-source Spectral linter; for more information about Spectral, see https://docs.stoplight.io/docs/spectral/674b27b261c3c-overview.

You can complete this task by using the API Manager UI, and by using API Designer when the UI is online, and is connected to a cloud instance that has the API governance microservice enabled on it. You can also use the toolkit CLI; for details see Validating an API document by using the toolkit CLI.

The following methods can be used to validate your API documents with rulesets by using the API Manager or the API Designer UIs:

- From within an API
- From within the Rulesets tab

Procedure

- To validate an API document from within the API, complete the following steps.
  1. Open the required API for validating, as described in Editing an OpenAPI 2.0 API definition or Editing an OpenAPI 3.0 API definition.
  2. Click Validate > With rulesets from the menu bar in the header.

  3. Select the rulesets that you want to validate your API document against, and click Next.
  4. Select the rules from within the rulesets that you want to use for the validation. By default, all of the rules are selected.
  5. Click Validate.
     If any validation errors are found, a scorecard is displayed showing the results of the validation. You can click on an error to take you to the specific section of the API that needs correcting. The API can then be updated to resolve the validation errors, and validated again. If no errors are found, a success message is displayed.
  6. Optional: You can click Download to download the results of the validation into a .csv file.
  7. Optional: You can click Back to change the rulesets and rules, and then run the validation again.
  8. Click Done when finished.
- To validate one or more draft API documents from within the Rulesets tab, complete the following steps.
  1. In the API Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. Click Validate.

4. Select the rulesets that you want to validate your API document against, and click Next.
5. Select the rules from within the rulesets that you want to use for the validation. By default, all of the rules are selected.
6. Select the APIs that you want to validate, and click Validate.
   If any validation errors are found, a scorecard is displayed showing the results of the validation. You can click on an error to take you to the specific section of the API that needs correcting. If no errors are found, a success message is displayed.
7. Optional: You can click Download to download the results of the validation into a .csv file.
8. Optional: You can click Back to change the rulesets, rules, and APIs, and then run the validation again.
9. Click Done when finished.

Results

Your API document is validated against one or more rulesets.

What to do next

You can use the results of the validation to help you improve your API document. You can also use the results to help you edit the rulesets, and create new rulesets; see Configuring API governance in the API Manager for more information.

- **Validating an API document by using the toolkit CLI**
  How to use the API Connect toolkit CLI to run API governance validation scans on your API documents.

Related information

- The API governance ruleset lifecycle

Validating an API document by using the toolkit CLI

How to use the API Connect toolkit CLI to run API governance validation scans on your API documents.

About

API governance is an IBM® API Connect service that can be used to validate and enforce organizational governance policies and best practices to your API development process. The following information provides details of the toolkit CLI commands for running and managing validation scans on your API documents. You can also validate an API document by using the API Manager and the API Designer UIs. For more information about validation scanning, see Validating an API document by using API governance.

The following methods can be used to validate your API documents with rulesets by using the toolkit CLI:

- Validate a single API document that is stored locally
- Validate one or more draft APIs that are stored on the management server

Validate a single API document that is stored locally

To validate a single API document against one or more rulesets, complete the following steps. The OpenAPI document must be stored locally.

1. Log in to the toolkit as the owner or an admin of your provider organization.
   For example:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Run the **compliance:validate** toolkit command in `governance` mode.

   ```
   apic --mode governance compliance:validate --org <target_org> --server <server_url> --rulesets <ruleset_name_1>,
   <ruleset_name_2> API_FILE
   ```

   where:
   - `<target_org>` is the name of the provider organization that owns the APIs to be validated.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.

- `<ruleset_name_1>,<ruleset_name_2>` is a comma-delimited list specifying the names of the rulesets to be applied to the API. (Specify each ruleset name; do not use the ruleset IDs.)
- `API_FILE` is the path and filename of the OpenAPI document to be validated.

Validate one or more draft APIs that are stored on the management server

The APIs must be stored as draft APIs in your provider organization; this operation validates all of the draft APIs that are listed in the `COMPLIANCE_REQUEST_FILE`.

1. Create a JSON `COMPLIANCE_REQUEST_FILE` that specifies the location of the draft APIs and the rulesets to use during validation.
   The following example shows a configuration file that specifies two draft APIs, and two rulesets. In the `spectral-oas` ruleset, the `info-contact` rule is disabled and will be ignored during validation.

```
{
  "apiUrls" : ["https://example.com/api/orgs/768a5f80-177c-4100-b2b6-57ce6c4bf761/drafts/draft-apis/a08898fe-b422-41d3-b058-c3280d632e93","https://example.com/api/orgs/768a5f80-177c-4100-b2b6-57ce6c4bf761/drafts/draft-apis/26692eab-f43b-4af3-985c-2deaa401eb7f"]
  "config" : {
    "rulesets": [
      {
        "rulesetName": "spectral-oas",
        "disabled": ["info-contact"]
      },
      {
        "rulesetName": "spectral-owasp",
        "disabled": []
      }
      ]
 }
}
```

   where:
   - `apiUrls` is a comma-delimited list of URLs. Each URL specifies the location of the draft APIs in a particular provider organization.
   - `rulesetName` is a list of rulesets to be applied to the APIs during validation.
     For each ruleset in the list, you can optionally add the `disabled` property with a comma-delimited list of rules to ignore during validation.

2. Log in to the toolkit as the owner or an admin of your provider organization. See Step 1 for details.
3. Run the **compliance:validate_all** toolkit command in `governance mode`.

```
apic --mode governance compliance:validate_all --org <target_org> --server <server_url> COMPLIANCE_REQUEST_FILE
```

   where:
   - `<target_org>` is the name of the provider organization that owns catalog containing the APIs to be validated.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `COMPLIANCE_REQUEST_FILE` is the path and filename of the configuration file that lists the APIs and rulesets to be used during the validation (for example, validate-apis.json).

LDAP authentication

The Lightweight Directory Access Protocol (LDAP) is an internet protocol for accessing and maintaining distributed directory information services over a network. If you rely on LDAP to authenticate users for web applications, take a minute to review the contents of this topic before beginning.

Programming guidelines

When authenticating with LDAP, observe the following guidelines:

- Use only a LDAP interface -- Users can only be authenticated via a connection to an LDAP server.
- Active Directory -- Use of an Active Directory interface is prohibited, however, LDAP authentication with Active Directory is supported.
- Administrator properties -- The LDAP administrator must have access to all user's LDAP properties.

LDAP attributes read by IBM® API Connect are as follows:

Attribute	Definition
`mail`	User's SMTP email address.
`cn`	Used internally to determine user's common name.
`sn`	Used internally to determine user's last name or surname.
`givenname`	Used internally to determine user's first name.
Prefix from `Search dn`	Used as username.

Using an LDAP Server for User Authentication

Every entry in an LDAP directory server has a distinguished name (DN). The DN is the name that uniquely identifies an entry in the directory. A DN is made up of `attribute=value` pairs, separated by commas. For example:

```
cn=Ben Gray,ou=editing,o=New York Times,c=US
cn=Lucille White,ou=editing,o=New York Times,c=US
cn=Tom Brown,ou=reporting,o=New York Times,c=US
```

Any of the attributes defined in the directory schema can be used to make up a DN. The order of the component attribute value pairs is important. The DN contains one component for each level of the directory hierarchy from the root down to the level where the entry resides. LDAP DNs begin with the most specific attribute (usually some sort of name) and continue with progressively broader attributes, often ending with a country attribute. The first component of the DN is referred to as the Relative

Distinguished Name (RDN). It identifies an entry distinctly from any other entries that have the same parent. In the previous examples, the RDN "cn=Ben Gray" separates the first entry from the second entry, (with RDN "cn=Lucille White"). These two example DNs are otherwise equivalent. The attribute=value pair making up the RDN for an entry must also be present in the entry. (This is not true of the other components of the DN.)

Examples

In the following examples, a user search is performed from Base DN. The search is done anonymously, or if authenticated bind is used, an authenticated user is used. Search DN appends the prefix, the given user name, and the suffix. If the prefix used is (**uid=** and the suffix used is "**)**", **uid** becomes the user name attribute. The default search filter used is: **(|(cn={filter}*)(sn={filter}*)(mail={filter}*)(givenName={filter}))** and attributes used for prefix are also added to the search filter. In this case, where '(**uid=**' is the prefix, when searching for users, the search filter becomes:

```
  (|(cn={filter}*)(sn={filter}*)(mail={filter}*)(givenName={filter}*)(uid={filter}*))
```

where **{filter}** is replaced by actual text.
The authenticated bind DN is a user on the external LDAP server permitted to get base DNs and search the LDAP directory within the defined search base. It should also be able to read other user properties and be used if anonymous access to LDAP to get base DNs and to search and get access to user attributes is not allowed. When a search is performed for *Steve*, the LDAP query filter shown in the following example is used and search is done from base DN specified in UI. When the user's DN is returned, the DN and password are used to authenticate the user:

```
(|(cn=Steve*)(sn=Steve*)(mail=Steve*)(givenName=Steve*)(uid=Steve*)).
```

For bind during login calls, the search string used is the prefix. For example, if the prefix is '(**uid=**', the search string used to search for a user during log in becomes: (**uid=Steve**).

Using the mail attribute as the user name

When you want to use an email address as the user name, (for example *steve@company.com*), you typically use the mail attribute as the prefix '(**mail=**'. In this case, use the following search string to perform the search internally (assuming '(**mail=**' as prefix):

```
(|(cn=steve@company.com*)(sn=steve@company.com*)(mail=steve@company.com*)
(givenName=steve@company.com*)(mail=steve@company.com*))
```

In the previous example, if DN is found, the password is used with the DN to perform a bind. The first result is taken, so be sure to use a unique attribute for username. Next, the LDAP properties for the user are read (**cn**, **sn**, **mail**, **givenName**). If LDAP properties cannot be read using the logged-in user, they are read from the user's DN using authenticated bind credentials. If the user name attribute differs, it is also queried for. For example, if the prefix is '(**uid=**', the **uid** attribute is also read from the user's DN object:

```
(|(cn={filter}*)(sn={filter}*)(mail={filter}*)(givenName={filter}*))
```

Note: The prefix and suffix cannot be used to get a user's DN directly. For example, the following attempt to directly get a user's DN fails:

```
Prefix: "uid="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_con_ldap_requirements_,
Suffix: ",ou=users,dc=company,dc=com"
```

LDAP referrals

LDAP referrals allow a directory tree to be partitioned and distributed between multiple LDAP servers. An LDAP referral is a domain controller's way of indicating to a client application that it does not have a copy of a requested object, while giving the client a location that is more likely to hold the object. The client then uses the object as the basis for a DNS search for a domain controller.

API Connect support for LDAP referral includes:

- Searching for users that are part of multiple Active Directory trees and forests.
- Authenticating users in the Cloud Manager, API Manager and the Developer Portal.

Note: API Connect LDAP referral support is dependent on the following conditions:

- A single LDAP host/port is configured with administrator credentials and all users are referred to from the tree/server's base DN.
- As part of the user search following the LDAP referral , the same administrator credentials are used in the downstream trees/forests.
- LDAP API authentication does not support external LDAP referral. Internal LDAP referral is supported.

Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service to manage user authentication, and to optionally provide additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes **cn=spoon,o=eatery**. The credential is kept in the OAuth **access_token** to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the **access_token**.
- Overriding the **scope** that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the **scope** value that the application receives. For example, you can provide a specific resource owner an account number within the **scope** header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use **cn=spoon,o=eatery** as the user identity.

```
HTTP/1.1 200 OK
Server: example.org
```

`X-API-Authenticated-Credential: cn=spoon,o=eatery`

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see User Security policy.

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see Example - using multiple OAuth policies in an OAuth provider assembly.

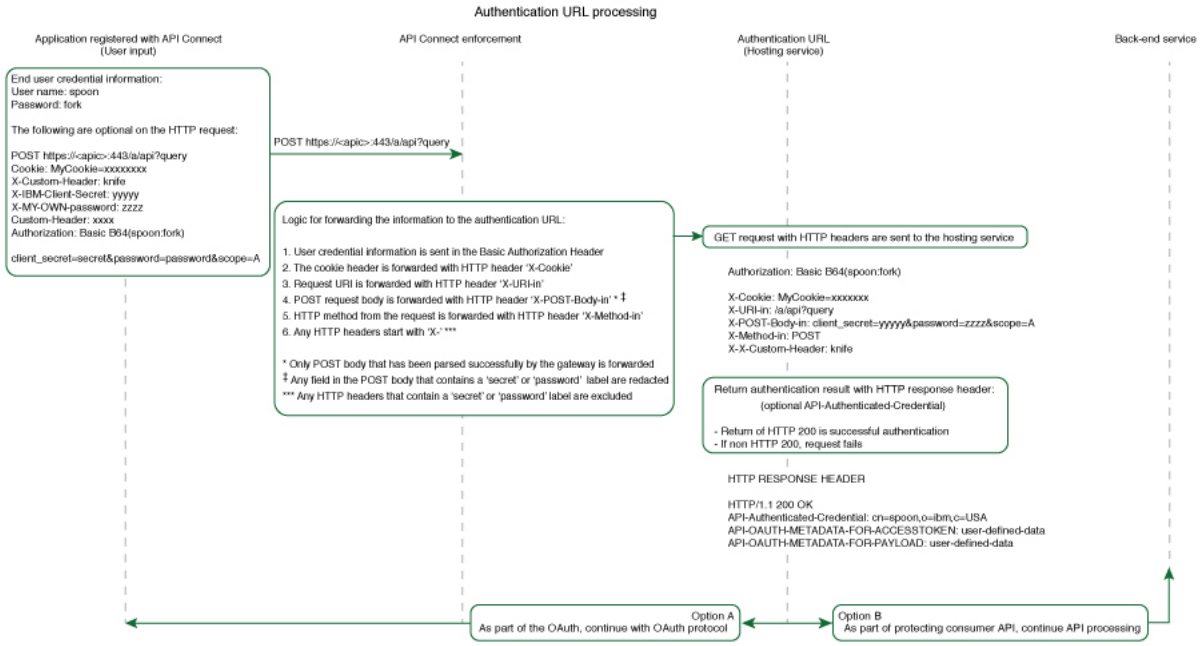
API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see OAuth external URL and authentication URL. The two metadata response headers are:

`API-OAUTH-METADATA-FOR-ACCESSTOKEN`
`API-OAUTH-METADATA-FOR-PAYLOAD`

When an Authentication URL is invoked, an HTTP response header is available to override the requested `scope` from the application. For more information, see Scope. The response header is:

`x-selected-scope`



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

Activating an API

After you have created an API definition you can activate it to auto-publish your API document and make it available for testing.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you activate an API, API Connect automatically completes the following actions:

- Creates a draft Product, adds the API to the Product, and publishes the Product to the Sandbox Catalog so that the API is available to be called. The Product has the title *api_title* auto product. Note that if you later want to delete the draft Product, you cannot delete it directly; instead, delete the API and the draft Product is deleted together with the API; see Deleting an API definition. If you want to remove the Product from any Catalogs to which it is published, you must do this separately; see Removing a Product from a Catalog
- Subscribes the Sandbox test application to the Product so that you can immediately test the API in the test environment. For information on testing an API, see Testing an API.

To activate an API, you must be assigned a role that has the `Product:Manage` and `Subscription:Manage` permissions. The pre-supplied Developer role has these permissions by default; if you are assigned a custom role it must have these permissions. For more information, see Creating custom roles.

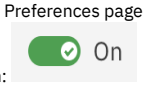
Note:

- The API activation will not complete successfully if lifecycle approval is enabled in the Sandbox Catalog for the Stage, Publish, or Replace actions. If any such lifecycle approvals are enabled, then to be able to activate an API they must be disabled; for information on lifecycle approval settings, see Creating and configuring Catalogs.
- To activate an API from the API Designer user interface, you must be connected to a Management server; API activation is not available with API Designer in offline mode. For more information, see Logging in from API Designer.
- Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "`(?`" cannot be validated or published. An error message is returned. For example:

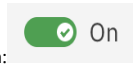
```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
    definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0,
col: 0)
400
```

Procedure

To activate an API and auto-publish your API document, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Under Testtab, click Target Configuration. The Preferences page opens.
4. Set the Auto-publish toggle to the On position: 

Results

On successful completion, the API is shown as On: 

You can stop the API by moving the activation slider control to the Off position:

If you stop an API, the application subscription is deleted, and the auto Product is removed from the Sandbox Catalog.
Note: You can select a Catalog of your choice, or you can opt to use the Sandbox Catalog by selecting **Use the default built-in Sandbox Catalog**. For more information, see Specifying the testing preferences for an API.
If you make a change to the API, it is republished automatically, You can also republish a running API manually by stopping it and then re-activating it.

The error indicator shows whether there are validation errors in the OpenAPI source for the API definition. If there are errors, click the icon for more details:

You can also activate an API during the creation process, and on the API test page; see Creating an API definition and Testing an API.

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

Testing an API

Select a test tool and verify that your API works as you intended.

Before you publish an API where customers can access it, you need to test it and ensure that it is defined and implemented correctly. API Connect offers tools for running both simple and complex tests, in different environments. Use the following list of options to choose the test tool that best meets your needs:

- Execute and debug the API in the Test tab
  On the Test tab, you can quickly invoke your API from the API Connect UI with minimal preparation. The Test tab offers full control over the API call through a simple set of fields where you can define your request. The response includes both a visual flow diagram and a code trace that shows how the API's process flow is executed. Note: the Test tab tool is only available to APIs that use the DataPower API Gateway.
- Call the API externally from the Automated API behavior testing application as explained in Testing an API with Automated API behavior testing
  Automated API behavior testing provides a simple set of fields where you can quickly invoke your API. You can automate tests and define schedules for running them. You can even set up API monitoring and receive alerts on your API health before issues occur.
- Call the API from a local test environment
  The Local Test Environment is a lightweight API Manager, running on your computer, that allows you to rapidly test APIs without connecting to an API Connect management server.
- Run a simple test in the Policies editor
  Note: The Test panel is deprecated and might be removed without warning.
- Use the Try it tool in the Explorer tab to run simple tests
  You can use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can check the descriptions of the different artifacts, and review any schemas or examples, and you can also use the Try it tool to test the behavior of the API.

Note: Due to CORS, when a test call is made with a browser-based API test tool, an HTTP OPTIONS call is made to the API first. The OPTIONS call:

- Does not use any credentials, so is not attributed to any consumer organization.
- Is logged as an API event in the analytics data. Because the OPTIONS call does not use credentials, the API events are only visible in analytics data in the Cloud Manager and API Manager UIs. The OPTIONS events are not visible in the developer portal analytics view, which includes only analytics data for a single consumer organization.
- Is not included in consumer rate limits.
- Is included in consumption-based licensing (if used).

- **Locating API information on the Endpoints tab**
  The assembly tool's Endpoints tab opens a page that displays information for use when you call APIs externally. This tab displays only when an API's status is Online (the API was activated).
- **Using the Test tab to debug your API**
  Use the Test tab to execute calls and trace the API's actions in API Connect.
- **Testing an API with Automated API behavior testing**
  Use the IBM Automated API behavior testing application to invoke an API from outside of the API Connect Assembly page so that you can verify that it executes properly.
- **Testing an API with the Policies editor**
  IBM API Connect provides a basic test environment in the Policies editor so that you can ensure that your APIs are defined and implemented correctly.
- **Testing an API with the Local Test Environment**
  Use the Local Test Environment to test APIs on your local machine, without the need to connect to an API Connect management server. The Local Test Environment is a lightweight API Manager running on your local machine. It allows you to rapidly test APIs locally.
- **Testing an API with the Explorer tab**
  Use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can also use the Try it tool to test the behavior of the API.

Locating API information on the Endpoints tab

The assembly tool's Endpoints tab opens a page that displays information for use when you call APIs externally. This tab displays only when an API's status is Online (the API was activated).

Use the Endpoints tab to retrieve information that you can use when you call the API from an application.

Note: The fields in the Sandbox Sample Application Credentials section apply to the assembly tool's built-in client application and not to any application that you created. The credentials that display here are the default credentials that display in the assembly tool Test panel's "Identification" section.

To copy information from the Endpoints tab, click next to a field.

The following values for the API are displayed in the Endpoints tab so that you can copy them and paste them into calls:

Base API Endpoint
: The URL representing the endpoint of the API that you are editing and testing. To invoke the API, append one of its supported queries (path and parameters) to this URL.

Client ID and Client Secret
: The values of the built-in test application's client ID and client secret. You can invoke APIs externally by providing the application's credentials in the request header.

OAuth Token URL
: If you specified an OAuth provider in the API's security definition, then this field displays the URL where you can obtain an access token for calling the API.

OAuth Auth URL
: If you specified an OAuth provider in the API's security definition, then this field displays the URL where you can submit an access token and receive an authorization token in exchange. You then use the authorization token when calling the API.

Rate Limit
: If you specified a rate limit on the Security panel when you created the API, the limit is displayed here for reference. If you did not specify a rate limit, the default rate limit displays.

Using the Test tab to debug your API

Use the Test tab to execute calls and trace the API's actions in API Connect.

The Test tab not only lets you quickly invoke an API with any needed headers and parameters, but lets you view the content of the response in both parsed and raw format. Additionally, you can view a trace of the actions execute as the API's process flow operates to complete the call. Viewing the trace can help you debug problems with the API's execution.

Note: The Test tab tool is only available to APIs that use the DataPower API Gateway.
You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Complete the following tasks to prepare your API for testing, execute requests, and debug responses.

1. Preparing an API for debugging with the Test tab
   Ensure that your API's definition meets the requirements for debugging it in the API Connect Test tab.
2. Specifying the testing preferences for an API
   By default, when you test an API with the Test tab, a range of test parameters are preconfigured. For example, a default Product is automatically generated, to which the API is added, and that Product is published to the Sandbox Catalog. However, you can configure testing preferences for each of your APIs, including the required Product and target Catalog.
3. Sending the API request
   Fill in fields to set up the request for an API that you want to debug on the Test tab.
4. Reviewing the API response and trace
   Review the response to an API that was invoked in the API Connect Test tab. You can use the included trace information to debug the API's execution.
5. Testing a GraphQL API with the Test tab
   For a GraphQL API definition, the Test tab contains a GraphiQL editor. You can use the GraphiQL editor to help construct a GraphQL query. When you test your GraphQL API, the response is displayed in the GraphiQL editor.

Preparing an API for debugging with the Test tab

Ensure that your API's definition meets the requirements for debugging it in the API Connect Test tab.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

To invoke your API in the Test tab, the following requirements must be satisfied:

The DataPower API Gateway is configured to support the API probe
: To enable the Test tab's trace feature, the DataPower API Gateway must be configured to support gateway peering and the gateway probe. In a gateway service running in Kubernetes, the API probe is automatically enabled. In a gateway service running external to Kubernetes, the DataPower configuration must be added to the application domain supporting API Connect. Check with your administrator to confirm that gateway peering and the gateway probe are enabled as explained in Configuring DataPower API Gateway.

The API uses the DataPower API Gateway
: A gateway exposes APIs to calling applications, and provides processing actions that enable the APIs to integrate with various endpoints. API Connect supports two versions of the DataPower Gateway, but you can only debug APIs that use the DataPower API Gateway. The v5-compatible gateway is not supported. If your API definition has CORS enabled but you cannot see the Test tab, the API is probably configured to use the wrong version of the gateway.
You can determine which gateway your API uses by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click Develop, then select the APIs tab.
3. Click the title of the API you want to test.
4. On the API's Design page, locate the Gateway field and note which option is selected.

You can change the gateway selection for an API on the Design page, but you must ensure that the policies used in the API's process flow are compatible with the new selection. For information on changing the gateway for an API, see Specifying the gateway type for an API definition.

CORS is enabled in the API definition
: Cross-Origin Resource Sharing support ensures that the API can be accessed from another domain. CORS support must be enabled for your API to be properly invoked and traced. If your API uses the API Gateway but you cannot open the Test tab, it's possible that CORS was not enabled in the API definition.
You can enable CORS for you API by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click Develop.
3. On the Develop page, click the name of the API you want to test. The API's Design page appears.
4. Under the Gateway tab, locate the CORS field and select it to enable support.
5. Click Save in the page header.

The API is published in the Sandbox catalog with the default Product/Plan, or a specified catalog and Product/Plan
: By default, when you test an API with the Test tab, a default Product/Plan is generated for the API and published to the built-in Sandbox Catalog. However, you can optionally specify a range of API testing preferences including an alternative Product/Plan, and target Catalog; for more information, see Specifying the testing preferences for an API. When you publish (activate) an API, the API is enabled online and becomes available for use. You cannot execute an offline API, even for testing purposes. You can set an API online by completing the following steps:

1. Log in to API Manager.
2. In the navigation list, click Develop.
3. On the Develop page, click the name of the API you want to test.
4. In the Design page header, click the settings icon . The **Preferences** page appears.
5. Set the Auto-publish toggle to On position.

Next topic: Specifying the testing preferences for an API

Specifying the testing preferences for an API

By default, when you test an API with the Test tab, a range of test parameters are preconfigured. For example, a default Product is automatically generated, to which the API is added, and that Product is published to the Sandbox Catalog. However, you can configure testing preferences for each of your APIs, including the required Product and target Catalog.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note: This feature is not available if you are using the Local Test Environment.

Procedure

1. Open the required API for editing, in either of the following ways:
   - During the initial creation of an API, the API wizard guides you to enter the minimum configuration settings; on completion of the initial configuration, click Edit API.
   - To open an existing API for editing, complete the following steps:
     - In the navigation pane, click Develop, then select the APIs tab.
     - Click the title of the API version that you want to work with.
2. Under the Test tab, click Target Configuration.
   The Preferences page opens.
3. Use the Edit links to configure the following testing preferences, as required:

   Auto-publish

In order to execute an API for testing purposes, you must make it online. To make the API online and auto-publish your API document, set the Auto-publish toggle to the On position.

Target catalog
: The Catalog to which you want to publish the Product that contains the API. The selected Catalog is used as the scope for the other preference settings. You can select a Catalog of your choice, or you can opt to use the Sandbox Catalog by selecting Use the default built-in Sandbox Catalog.

Target Gateway service
: The gateway service to which you want to publish the Product that contains the API.

Target product
: The Product that will contain the API; you can select any of the Products from the selected Target catalog, or you can opt to use an automatically generated Product by selecting Generate Product on Save. The auto-generated Product has the title *api_title* auto product.

Target product rate limit
: If you opted to use an automatically generated Product, you can configure a rate limit to control calls to the API. If you selected your own Product, the Target product rate limit option is not available; instead, the rate limits that are configured in the selected Product and Plan are applied.

Target Plan
: The Plan in the selected Product that you want the test application to subscribe to. If you opted to use an automatically generated Product, the Target Plan option is not available; instead, the application is subscribed to the Default Plan in the automatically generated Product.

Test application
: The application that you want to subscribe to the selected Plan. You can select an application of your choice or, if you opted to use the Sandbox Catalog, you can instead use the pre-supplied test application by selecting Use the built-in Test Application.

Test Consumer Org
: The consumer organization that you want to subscribe to the selected Plan. If you selected your own application, the consumer organization is preset to that which contains the application, and cannot be changed. If you opted to use the pre-supplied test application you can, if required, select a consumer organization of your choice.
Click Reset to reset preferences to the default values.

4. Click Save Preferences when done.
   The Test page header displays the following details:
   - The target catalog.
   - The target gateway services.
   - The target product associated with the current API.
   - If the current status of API is Online or Offline. You can configure this in the API's Design page header.

Results

The next time you bring the API online, the API is published as specified by the preference settings, ready for testing.
Note: These settings are stored in your browser's local storage and will be used for your API testing in subsequent sessions. The following settings are specific to each individual API:

- Target product
- Target product rate limit
- Target Plan

The remaining settings will be applied to all your APIs.
Previous topic: Preparing an API for debugging with the Test tab
Next topic: Sending the API request

Sending the API request

Fill in fields to set up the request for an API that you want to debug on the Test tab.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Open the Test tab

1. Log in to API Manager or API Designer.
2. In the navigation list, click Develop, then select the APIs tab.
3. Click the title of the API you want to test.
4. On the API Design page, click the Test tab in the page header.

Fill in the "Request" section

Use the Request section of the page to set up the request URL, the authentication mechanism, and the request parameters. Complete the following steps to fill in the fields needed for configuring the request.

Note: The Test tab does not support vanity endpoints; you must specify the gateway endpoint at which the API is invoked. The **Request** field uses the gateway endpoint by default.

1. Select an operation and request URL from the list provided.
2. On the Parameters tab, define header, query and path parameters.
   An empty row is provided for you to define parameters; enter the parameter name in the Key field, select query, header, or path in the Located in field as appropriate, and supply a string value in the Value field. As you begin to define a new parameter, a further empty row is added automatically.

   Default header parameters and values are provided. For example, if the API has client ID or client secret definitions applied, the corresponding keys are added as header parameters, with the values preset; for details on configuring client ID and client security definitions, see Creating an API key security definition.

   The following table lists some common header definitions that are used in a request:

Table 1. Common request headers

Key	Value	Comments
Accepts	*media_type/sub type*	Specify the type of content that the response headers should use. The default is `application/json`.
X-IBM-Client-Id	*client_ID_val*	If you are using the built-in test application in the Sandbox catalog, the value is pre-populated automatically. If you are using your own client application, replace the value with the client ID of your application.
X-IBM-Client-Secret	*client_secret_val ue*	If you are using the built-in test application in the Sandbox catalog, the value is pre-populated automatically. If you are using your own client application, replace the value with the client secret of your application..
Authorization	Bearer *access_token*	Encode the token in base64. API Connect cannot encode the token for you.
Content-Type	*media_type/sub type*	Specify the type of content that the response body should use; for example, `application/json` or `image/png`.

If the API has path or query parameters defined, these are added to the parameters list, ready for you to supply values; for details on configuring path and query parameters, see Defining Paths for a REST API and Configuring an operation.

3. If your API definition uses Basic authorization for the security setting, click the Authentication tab and provide the User name and Password needed for authenticating with the user registry.
   When you invoke the API, API Connect populates a header using the provided information.

   If you API definition does not use Basic authorization, the Authentication tab is not available.

4. If the operation is POST or PUT, set up the request body.
   a. Click the Body tab.
   b. Type the information for the body of the request.

Send the request

When your API request is configured, click Send to execute the call.

If a message displays indicating "No Response", the API call cannot be completed. Possible causes of this problem are as follows::

- CORS is not enabled in the API's definition
  Edit the API's definition and enable CORS. Then, save the change and publish the API. If you are using your own client app, remember to subscribe it to the API again after publishing.
- The gateway service URL has an invalid certificate
  Follow the instructions to accept the certificate and continue testing.
- The browser cannot connect to the gateway service

Previous topic: Specifying the testing preferences for an API
Next topic: Reviewing the API response and trace

Reviewing the API response and trace

Review the response to an API that was invoked in the API Connect Test tab. You can use the included trace information to debug the API's execution.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When the API request is invoked, the response displays in the "Response" section of the Test tab. The response always includes the HTTP status code the amount of time it took to receive a response. If the call completes (even if it returns an error), the response also includes headers and a body.

Click the Body tab to see the body of the response. When you review the body of the response, click the Parsed tab to view the response using the format that you specified in the "Accepts" request header. Click the Raw tab to view the unformatted response body.

Click the Headers tab to see the content of request headers and response headers.
Attention: If the assembly or back-end API returns a Cache-Control header, add the `Cache-Control: no-store` request header to prevent the browser from caching the request. Depending on your server, you might want to use the `Cache-Control: no-cache` header instead.
Click the Trace tab to display a record of the API execution so you can see what actions were triggered and the code that we executed for each action.

Examine the trace

Use the Trace tab to see exactly how the API call was executed. This is helpful when the call returned an error and you don't know why. You can debug the API by reviewing the trace to see each step of the API's execution.

Note: The Trace works only with the default Gateway's URL. You cannot select a different Gateway for the trace.
The Trace tab contains the following components to help you debug the API:

Process flow diagram
: In the diagram, the policies (actions) that were executed during the call are highlighted while the rest of the process flow is dimmed. The highlights make it easy to see where the executed actions occurred in the overall flow while clarifying the actual flow of execution for the call. For example, if the process flow includes a switch with three options, only the option that was selected during the call is highlighted.

Policies list
: A list of the highlighted policies lets you select a policy to examine its trace.

Advanced toggle
: The Advanced toggle that lets you control the level of detail in the trace. A basic trace shows the code for the input and output of the selected policy. An advanced trace shows the full code for the policy's execution.

Code box
: The selected level of code for the current policy displays in the code box, where you can review it in detail to see exactly how that policy was executed. By default, a minimal version of the response displays (the endpoint, the request, and the response, and the status message), with sections collapsed. You can expand individual sections by clicking the expand icon next to each. To see the complete response, click the Advanced toggle.

Previous topic: Sending the API request
Next topic: Testing a GraphQL API with the Test tab

Testing a GraphQL API with the Test tab

For a GraphQL API definition, the Test tab contains a GraphiQL editor. You can use the GraphiQL editor to help construct a GraphQL query. When you test your GraphQL API, the response is displayed in the GraphiQL editor.

You can test a GraphQL query in either of the following ways:

- Use a **`POST`** operation, in which case the query is sent in the body of the operation request.
- Use a **`GET`** operation, in which case the query is sent as a query parameter.

For each operation type, you can test the query against either the GraphQL API endpoint itself, or against the cost endpoint that returns details of the cost of a request to the GraphQL API endpoint.

Whichever method and endpoint you use, you can either supply the query directly, or you can use the GraphiQL editor to help construct the query; the editor provides type ahead assistance based on the schema that is defined for the API, and highlights errors along with correction suggestions. The schema was initially uploaded when the GraphQL API definition was created, but might have been modified subsequently. For details on creating a GraphQL API definition, see Creating a GraphQL proxy API. For details on modifying the schema, see Using the GraphQL schema editor.

Before you begin

See Preparing an API for debugging with the Test tab for the requirements that your API definition must meet for you to test it with the Test tab.

Procedure

1. In the Request section, select the required operation type and endpoint; for example:
   - **`POST https://myserver.com/myorg/sandbox/mybasepath/graphql`**
   - **`GET https://myserver.com/myorg/sandbox/mybasepath/graphql/cost`**
2. On the GraphiQL tab, in the left hand pane of the editor, supply the query; for example:

```
{
  accounts(limit: 100) {
    name {
      first
      last
    }
  }
}
```

3. If required, you can supply additional request headers on the Parameters tab.
4. If your query includes variables, you can supply values for the variables in the Query Variables pane. For example, if the query is as follows:

```
query MyQuery($Query__accounts__limit: Int)
{
  accounts(limit: $Query__accounts__limit) {
    name {
      first
      last
    }
  }
}
```

   you can set a value for the **`Query__accounts__limit`** by entering the following in the Query Variables pane:

```
{
  "Query__accounts__limit": variable_value
}
```

   where *variable_value* is the required value.
5. Click either Send, or the Execute Query icon ▶ in the editor. The response is displayed in the GraphiQL editor, and the Trace section shows you how the API call was executed. For details on the trace information, see Examine the trace.

Previous topic: Reviewing the API response and trace

Testing an API with Automated API behavior testing

Use the IBM® Automated API behavior testing application to invoke an API from outside of the API Connect Assembly page so that you can verify that it executes properly.

Automated API behavior testing provides a simple set of fields where you can quickly invoke your API. You can automate tests and define schedules for running them.

Automated API behavior testing is an optional add-on to API Connect, and must be deployed by your administrator. If Automated API behavior testing is available in your deployment, a Test API displays on the Home page in API Manager.

Known limitations:

- AutoTest Assist is only available in Cloud Pak for Integration 2022.2.1 and later.
- Insights-Driven Test Generation is only available in Cloud Pak for Integration 2021.2.1 and later.

The following OpenAPI 3.0 features are not currently supported in Automated API behavior testing, but their usage doesn't impact on any other supported features:

- The `links` object; allows developers to describe how various values returned by one operation can be used as input for other operations.
- The `callbacks` object; enables asynchronous, out-of-band requests that your service sends to some other service in response to certain events.
- The `oneOf` keyword for defining complex schemas; allows developers to define more sophisticated validation rules. Automated API behavior testing doesn't generate a valid test for this keyword.

Note that you can add checks for missing assertions manually by using the `Add Request/Assertions` option.
If you want to use Automated API behavior testing for validating an API, complete the following steps to get started.

1. Log in to API Manager.
2. Publish your API to make sure that it's available online.
3. Create a client application and copy its client ID and client secret.
4. Subscribe the client app to the API.
5. Open the Test application.
    - Cloud Pak for Integration only: Click Design in the page header, and then click Test APIs; or
    - All users: Click API Manager in the banner to return to the Home page, and then click the Test API tile.

Getting started

The Automated API behavior testing interface provides collaborative and powerful API testing capabilities.

- **Create automated and simple to use API tests quickly and easily.**
    - Produce high quality APIs consistently, with automated and simple to use API testing that you can rely on.
- **Intelligent validation of API payload accuracy.**
    - Add more intelligence to your tests to verify the accuracy of the payload. You can easily make logical assertions to create flexible and powerful conditions for validation, with no code necessary.
- **Collaborative testing.**
    - Accelerate your API development and testing, by sharing your tests and results.
- **Easily integrate into your CI/CD pipeline to execute tests as part of your deployment.**
    - Create API Hooks to allow your tests to be run via the command line and become part of your automated build processes.
- **No code necessary.**

How to get started

Creating a test involves the following steps:

1. Creating a test suite
2. Creating test cases in a test suite
3. Defining variables
4. Running the test
5. Publishing the test

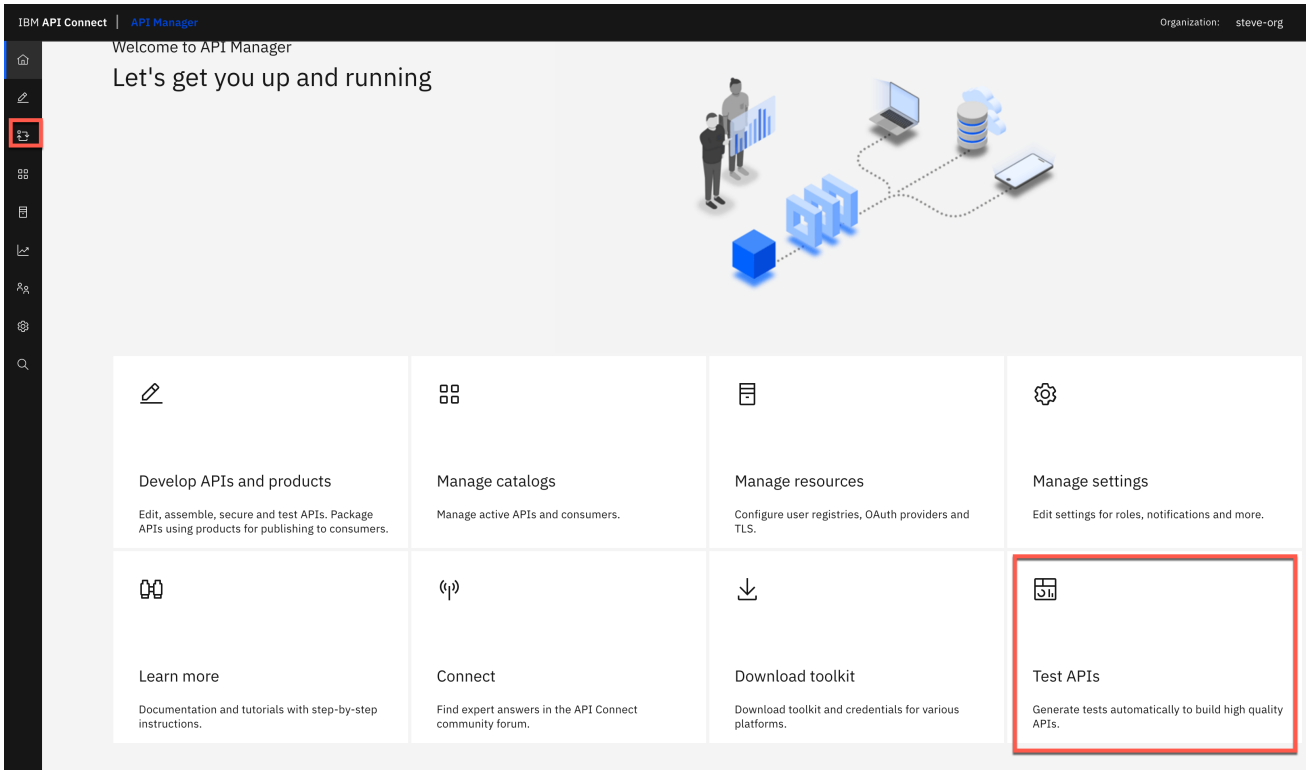
Explore the following tasks for detailed instructions:

- Creating a test suite
- Modifying a test suite
- Deleting a test suite
- Creating a test case
- Modifying a test case
- Defining variables
- Deleting a test case
- Running a test case
- Viewing the test report
- Publishing a test

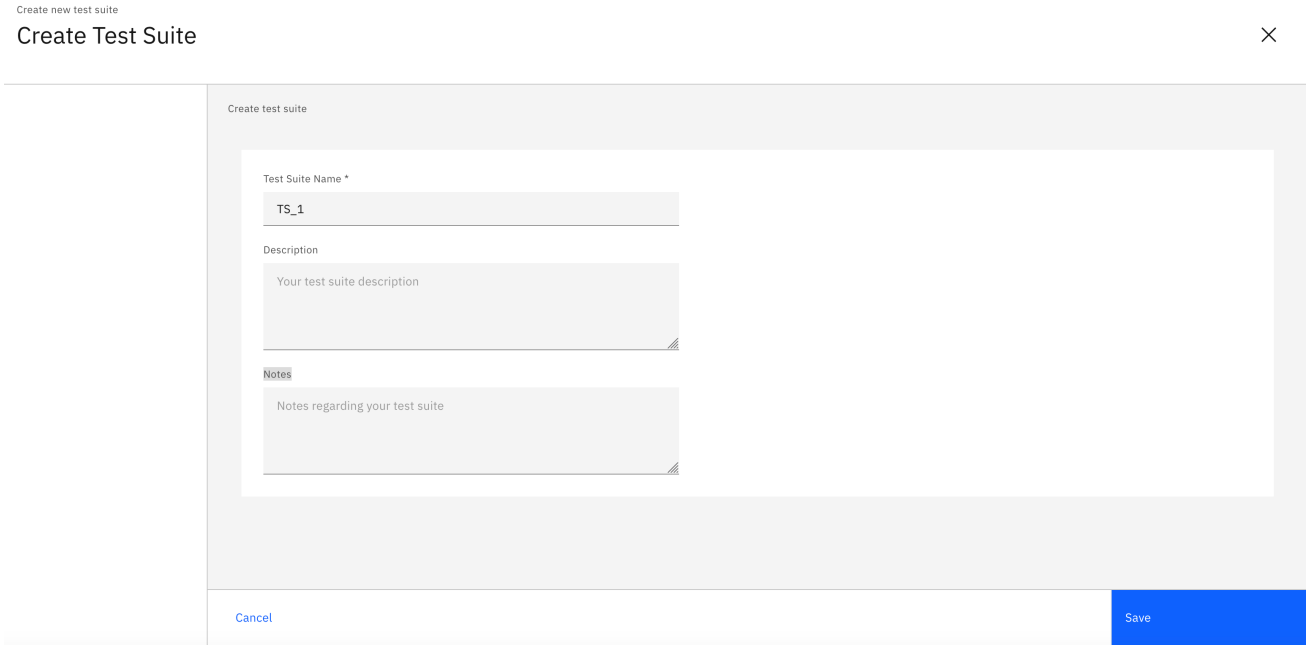
Creating a test suite

The following instructions show you how to create a test suite using the **API Manager**.

1. Log in to the API Manager.
2. On the Home page, click the **Test APIs** tile.



3. On the Test page, click **Add test suite**.



4. In the Create Test Suite page, enter a test suite name, description, and any relevant notes for the test suite.
5. Click **Save**. Once the test suite is created, you can add multiple test cases.

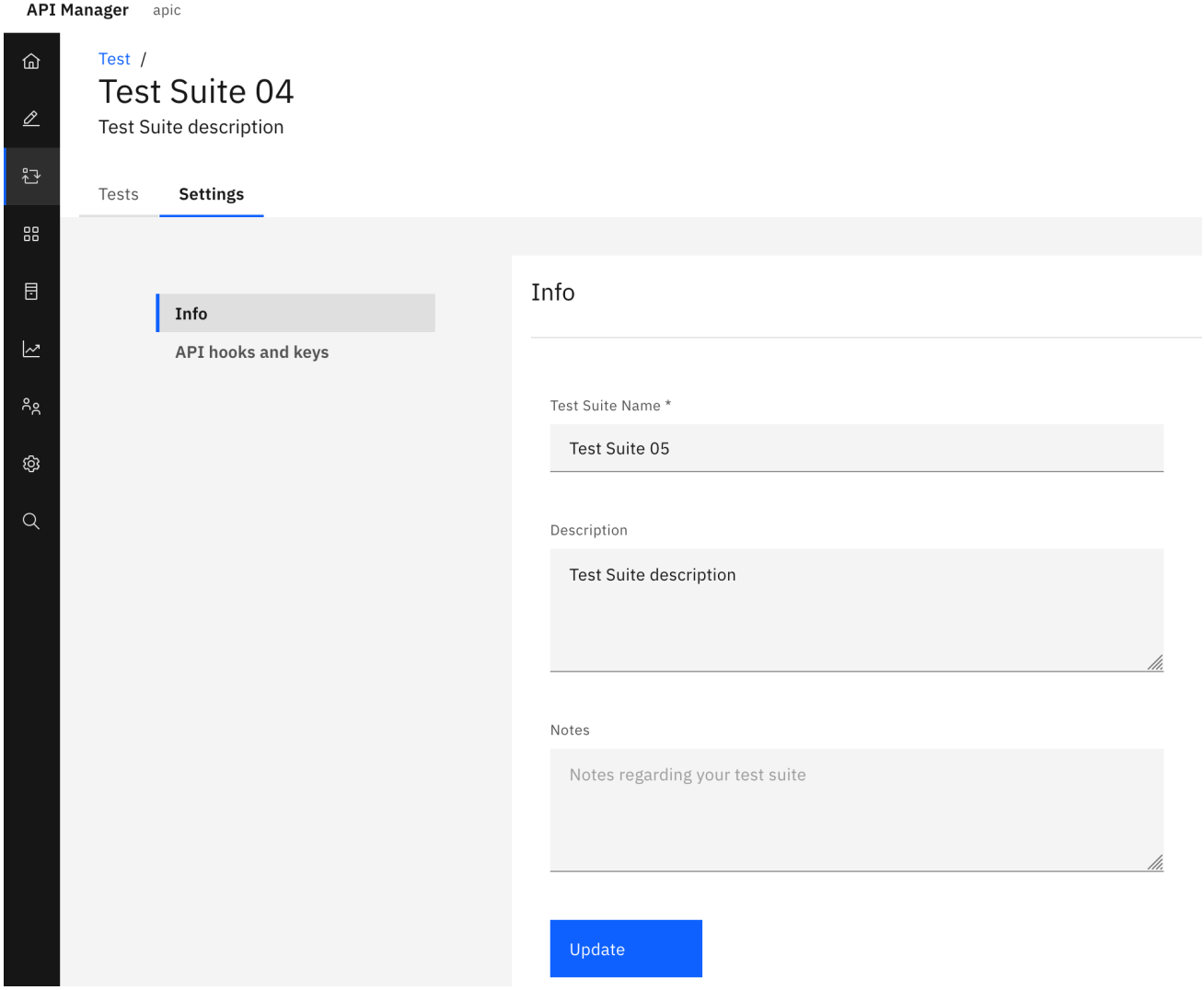
What to do next

- Next topic: Creating a test case

Modifying a test suite

The following instructions show you how to modify a test suite using the **API Manager**.

1. Log in to the API Manager.
2. Click the Test APIs tile.
3. Select a test suite to modify.
4. Go to the **Settings** tab.



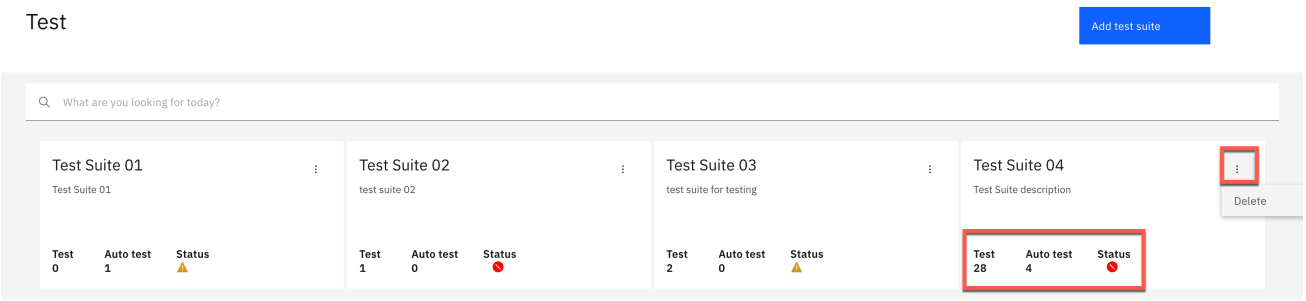
5. In the **Info** tab, update the following:
   - Test Suite Name
   - Description
   - Notes
6. Click **Update**.

- Next topic: Modify a test case

Deleting a test suite

To delete a test suite, complete the following steps:

1. Login to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, locate the test suite you want to delete and determine whether it contains any tests by observing the displayed **Test** count.



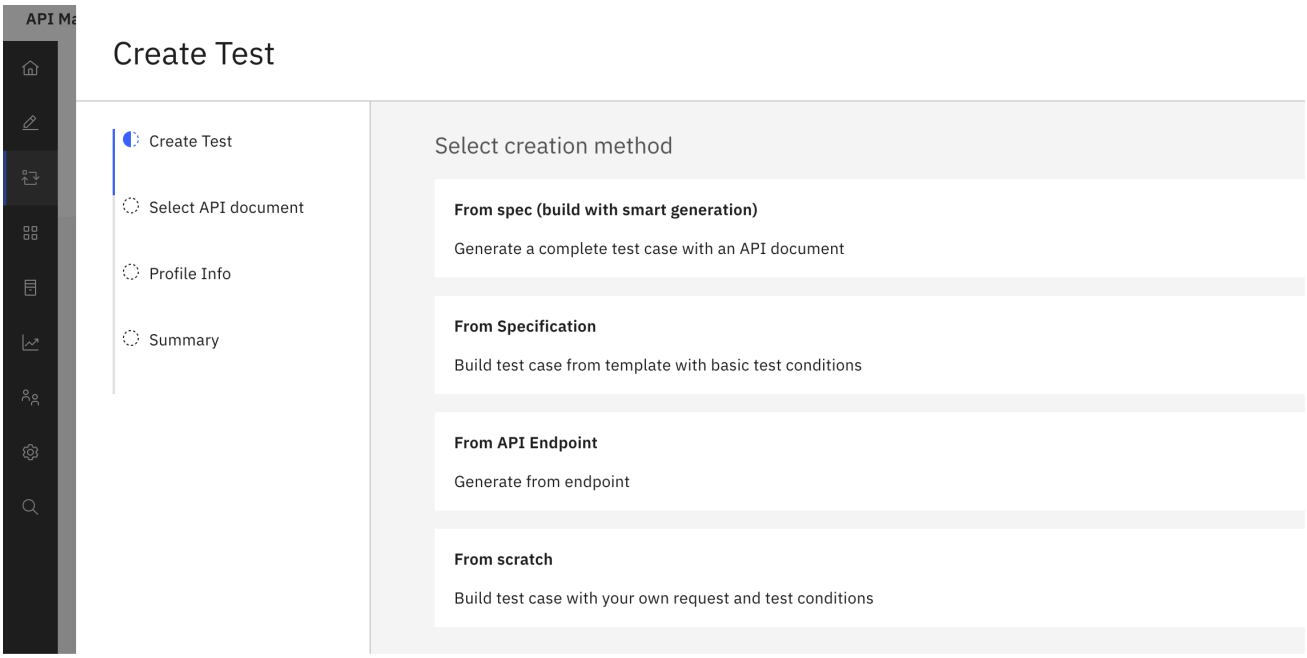
Note: You can't delete a test suite that contains tests, so if your test suite does contain any tests you must delete them first; see Deleting a test.

4. To delete the test suite, click its **Options** icon, then click **Delete**.

Creating a test case

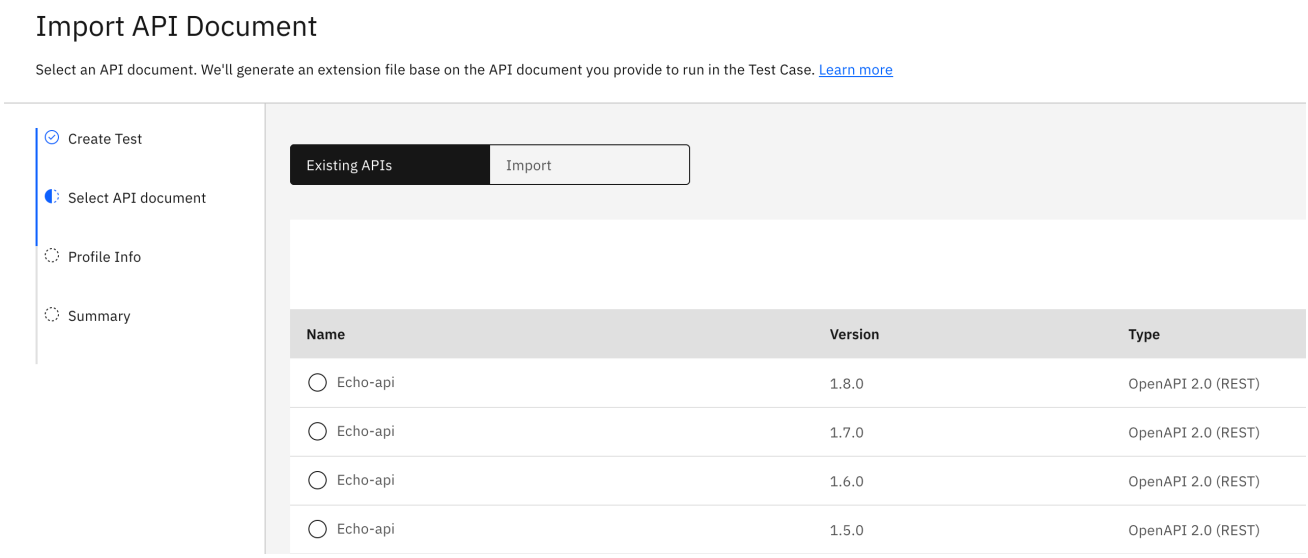
The following instructions show you how to create a test case using the **API Manager**.

1. Log in to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, select a test suite.
4. In the Create Test page, go to the **Tests > Test Cases** tab, and click **Add**.
5. Select a test case creation method:



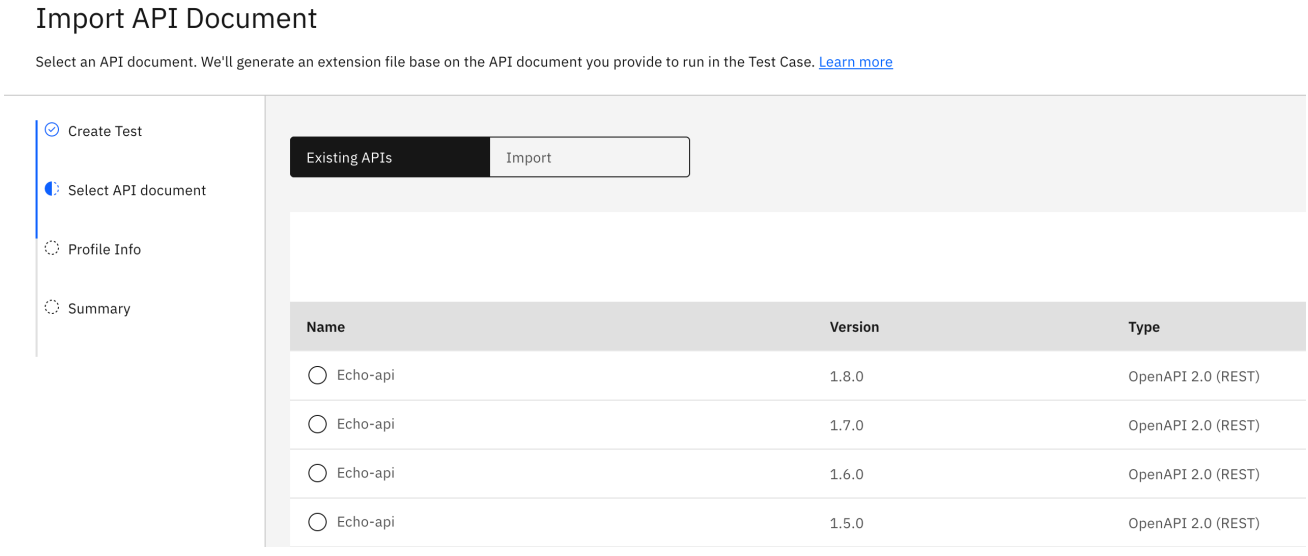
- From spec (build with smart generation): This feature is available only in *API Connect Enterprise as a Service.* The application generates a complete test case that leverages NLP to help developers and QAs test their APIs effectively in an efficient way.
- From Specification: Builds test case from OpenAPI specification
- From API Endpoint: Generates test case from a specified endpoint
- From scratch: Creates a test case with your own request and test conditions

6. If you selected **From spec (build with smart generation)**, click **Next**, then perform one of the following steps:



1. Go to the **Existing APIs** tab:
    1. Select a predefined API.
    2. Click **Next**.
    3. Select a path.
    4. Enter the API name.
    5. Provide a brief description.
    6. Add tags.
    7. Click **Next**. You will see the message that the test case has been generated.
    8. Click **Edit**. You will see the generated test assertions. You may edit the test assertions as needed.
    9. Click **Save**.
    10. Click **Run Test**.
2. Go to the **Import** tab, do one of the following:
    - Drag and drop a JSON or YAML file that has the API test case definition OR
    - Enter an URL for the API test case. Provide username, and password if required, for authentication.
    1. Click **Next**.
    2. Select a path.
    3. Enter the API name.
    4. Provide a brief description.
    5. Add tags.
    6. Click **Next**. You will see the message that the test case has been generated.
    7. Click **Edit**. You will see the generated test assertions. You may edit the test assertions as needed.
    8. Click **Save**.
    9. Click **Run Test**.

7. If you selected **From specification**, click **Next**, then perform one of the following steps:

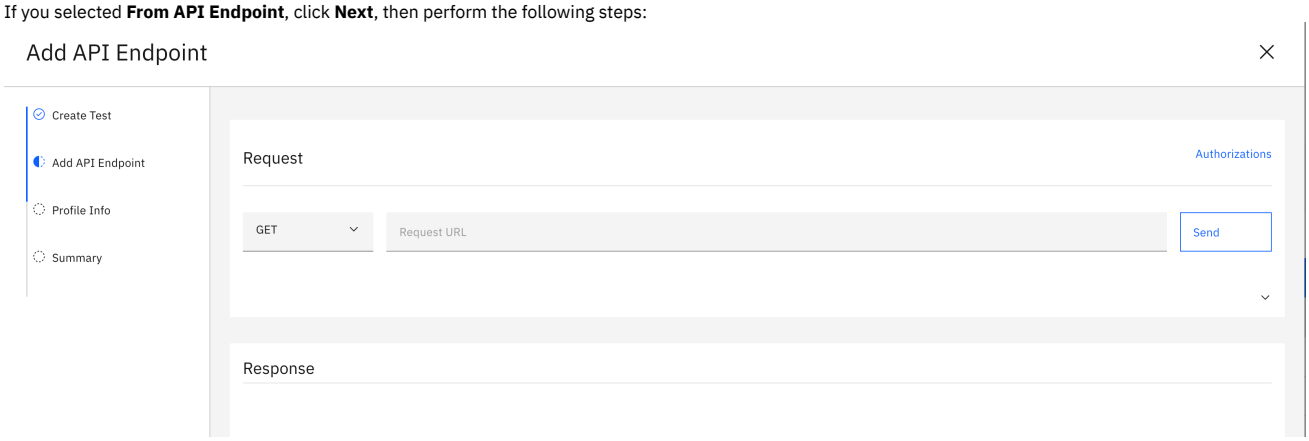


1. Go to the **Existing APIs** tab:
    1. Select a predefined API.
    2. Click **Next**.
    3. Select a path.

4. Enter the API name.
5. Provide a brief description.
6. Add tags.
7. Click **Next**. You will see the message that the test case has been generated.
8. Click **Edit**. You will see the generated test assertions. You may edit the test assertions as needed.
9. Click **Save**.
10. Click **Run Test**.

2. Go to the **Import** tab, do one of the following:
   - Drag and drop a JSON or YAML file that has the API test case definition OR
   - Enter an URL for the API test case. Provide username, and password if required, for authentication.
   1. Click **Next**.
   2. Select a path.
   3. Enter the API name.
   4. Provide a brief description.
   5. Add tags.
   6. Click **Next**. You will see the message that the test case has been generated.
   7. Click **Edit**. You will see the generated test assertions. You may edit the test assertions as needed.
   8. Click **Save**.
   9. Click **Run Test**.

8. If you selected **From API Endpoint**, click **Next**, then perform the following steps:



   1. In the Request section, select a method, enter the API endpoint that was generated when you published the API, set the authorization if required, and click **Send**. You will see the response in the **Response** section.
   2. Click **Next**.
   3. Enter the test name, description, and tags, then click **Next**. You will see the message that the test case has been generated.
   4. Click **Edit**. You will see the generated test assertions. You may edit the test assertions as needed.
   5. Click **Save**.
   6. Click **Run Test**.

9. If you selected **From scratch**, click **Next**, then perform the following steps:

   1. Enter the test name.
   2. Provide a brief description.
   3. Add tags.
   4. Click **Next**. You will see the message that the test case has been generated.
   5. Click **Edit**.
   6. Enter the API definition. Refer to the Test Defintion Syntax for a sample.
   7. Click **Save**.
   8. Click **Run Test**.

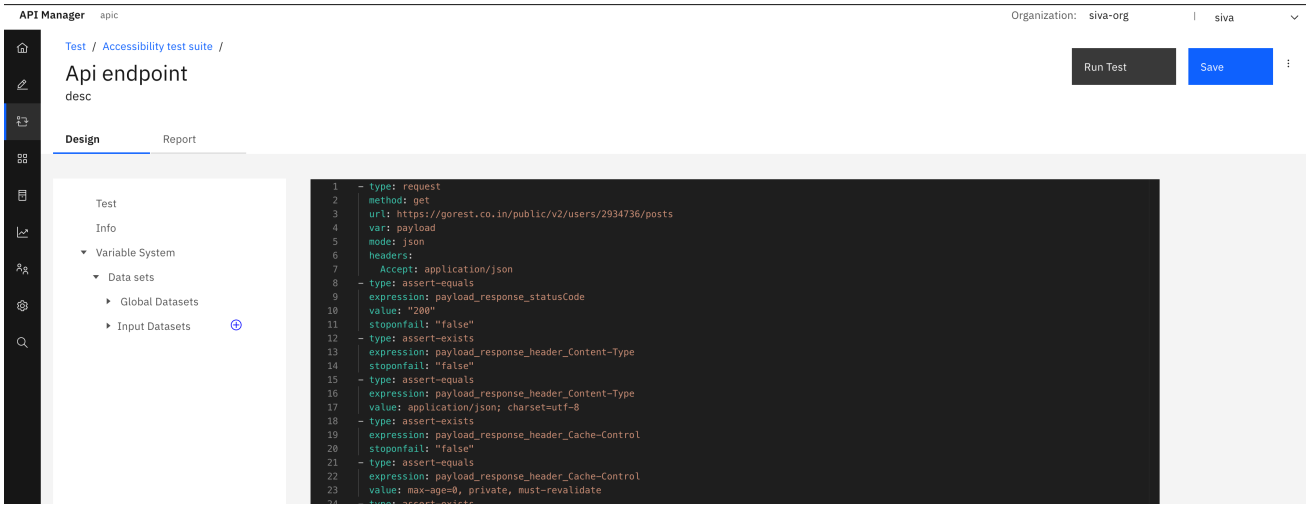
What to do next

- Next topic: Modifying a test case

Modifying a test case

The following instructions show you how to modify a test case using the **API Manager**.

1. Log in to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, click a test suite tile.
4. Select a test case you wish to edit.
5. Go to the **Design** > **Test** tab, modify the API definition as needed, and click **Save**.



6. Go to the **Info** tab, update the test case name, description, and tags, then click **Save**.
7. Click on **Variable System** > **Data Sets** > **Global Datasets** or **Input Datasets** tab, and update the global datasets or input datasets to add or modify the request payload and run the test case against each input datasets.

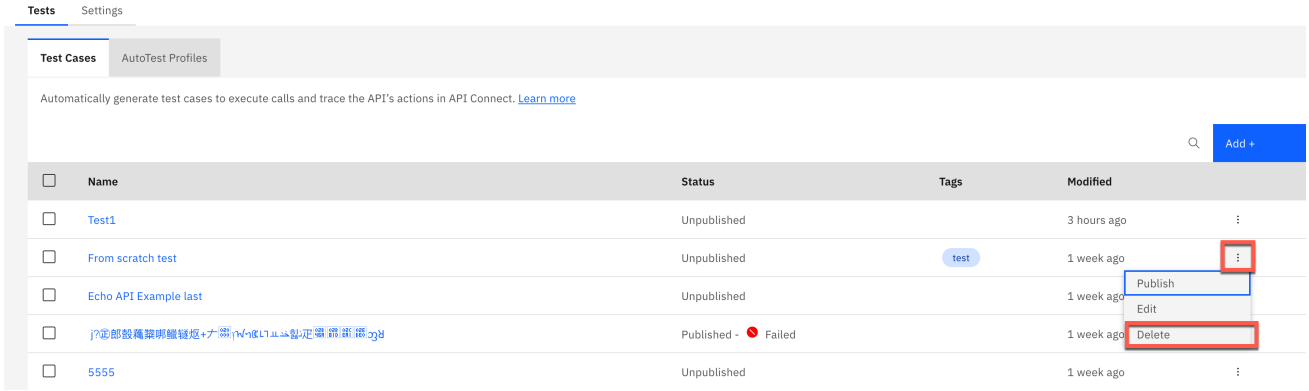
What to do next

- Next topic: Running a test case

Deleting a test case

To delete a test case, complete the following steps:

1. Login to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, select a test suite.
4. On the **Tests > Test Cases** tab, locate the test case you want to delete and click the **Options** icon > **Delete** option.



5. Click **Yes** for confirmation.

Defining variables

Within API Testing there are multiple ways to define variables to address different requirements.

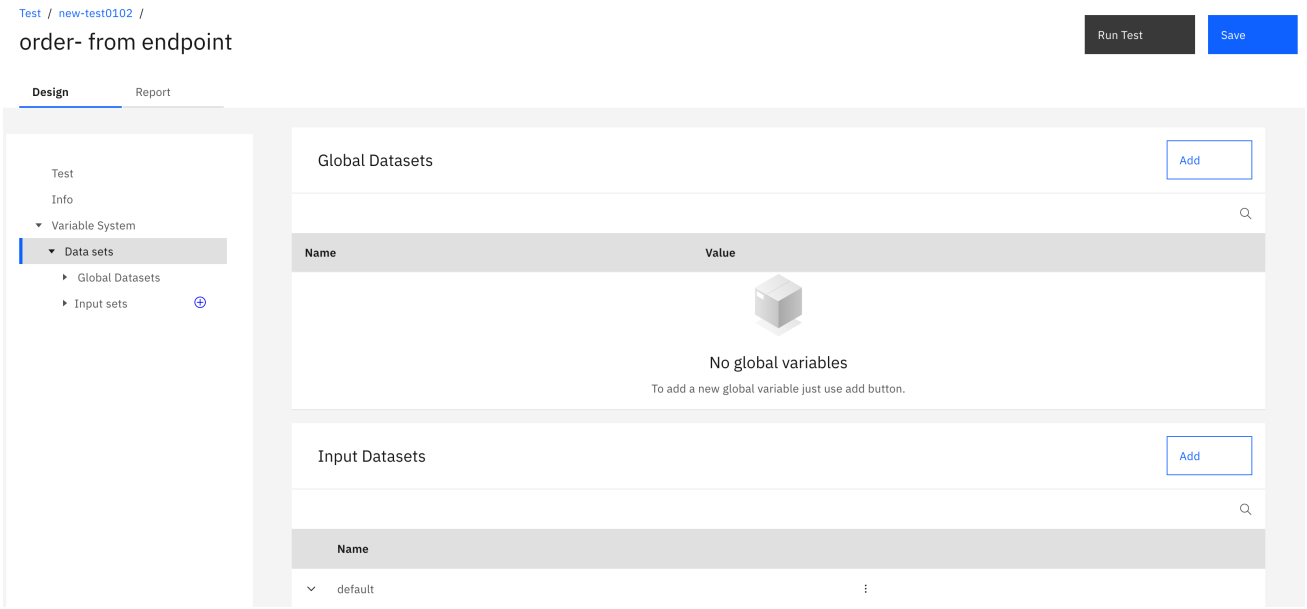
Global Datasets

Global Datasets are ones that are common across the whole test and typically does not change between runs based on Input Datasets

Adding a global variable

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite tile.
4. Select a test case.

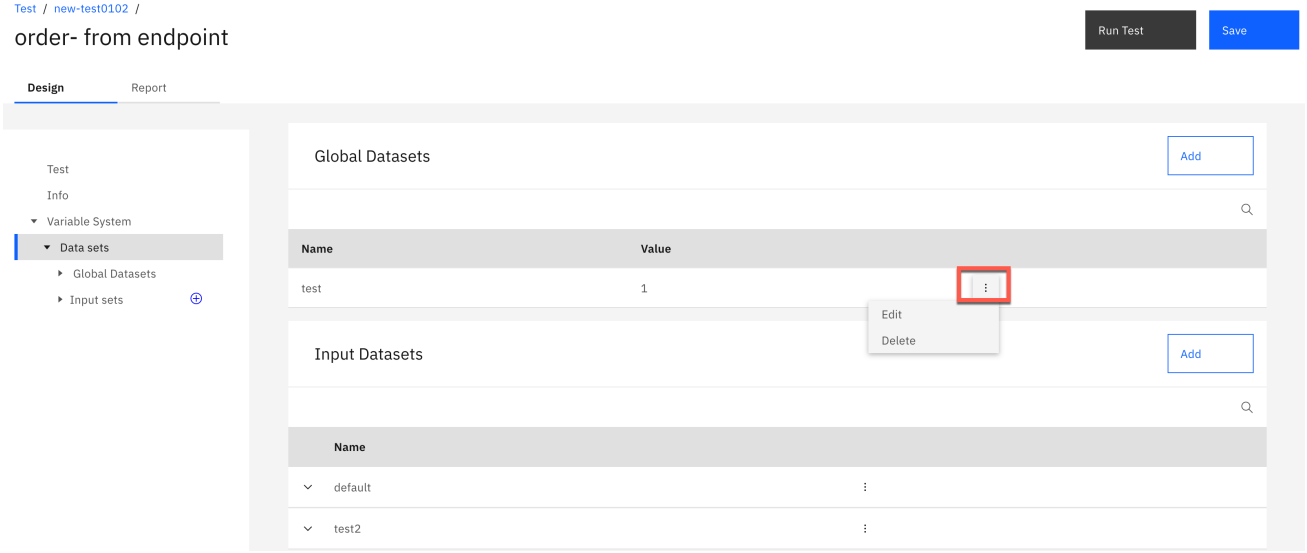
5. Go to the **Design** > **Variable System** > **Data sets** tab.



6. In the Global Datasets section, click **Add**.
7. Enter the variable name and value.
8. Click **Add**.

Editing or deleting a global variable

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite tile.
4. Select a test case.
5. Go to the **Design** > **Variable System** > **Data sets** tab.
6. In the Global Datasets section, click the options menu and select **Edit**.



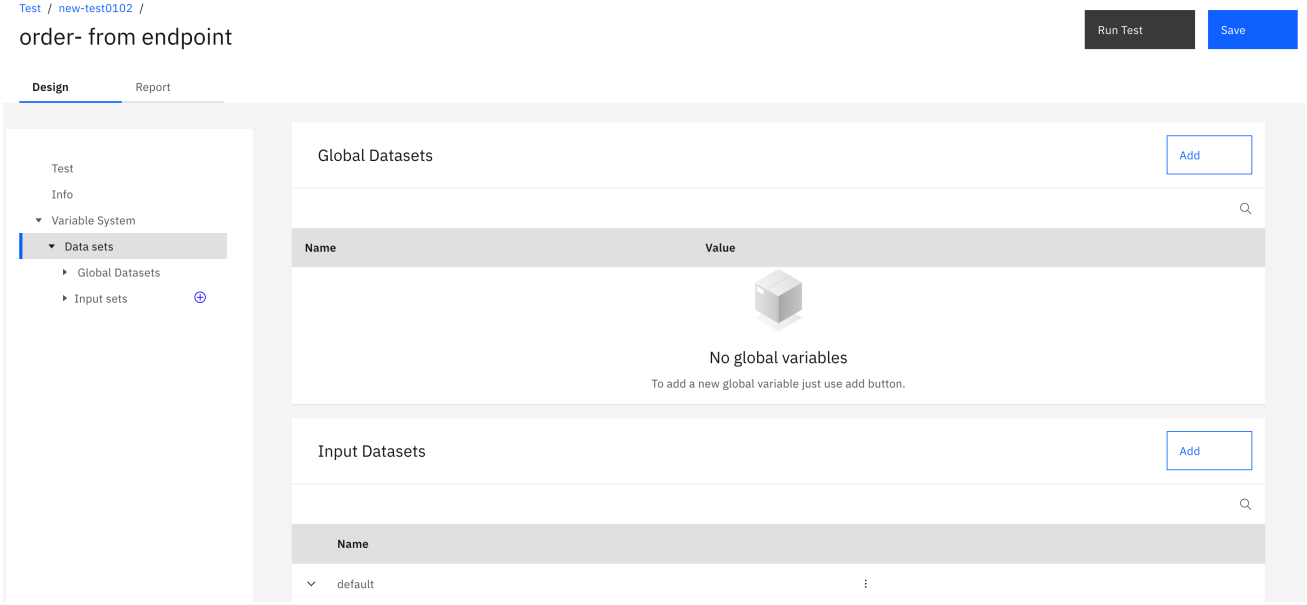
7. Update the name and value as needed.
8. Click **Add**.
9. To delete a variable, click the options menu and select **Delete**.

Input Datasets

The input datasets, is a group of input variables representing a scenario (e.g product id). Where the number of input datasets determines how many times the test will be executed using those values. If 3 input datasets are defined, the test will run 3 times using the inputs defined within each of those sets on each run.

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite tile.
4. Select a test case.

5. Go to the **Design** > **Variable System** > **Data sets** tab.



6. In the Input Datasets section, click **Add**.
7. Enter a dataset name.
8. Click **Add**.

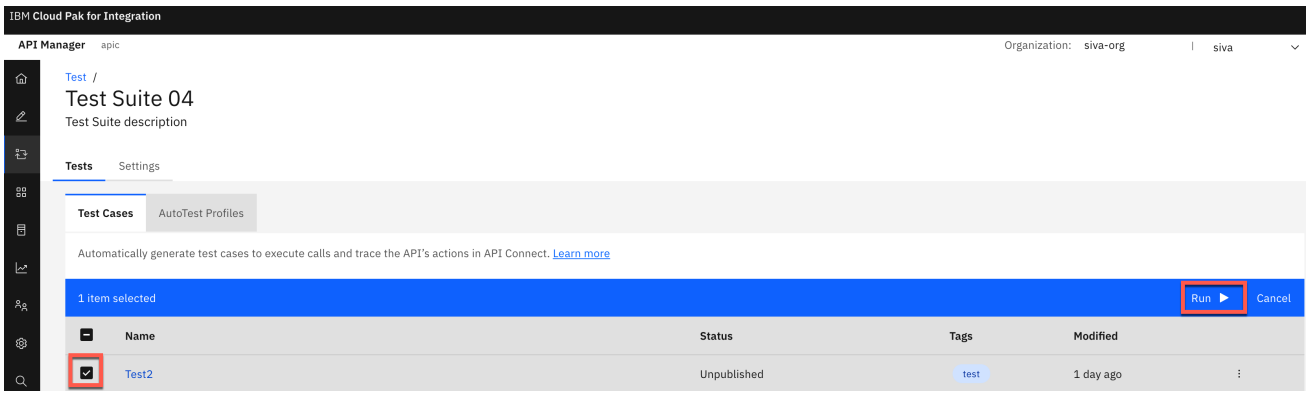
What to do next

- Next topic: Running a test case

Running a test case

The following instructions show you how to run a test case using the **API Manager**.

1. Log in to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, select a test suite tile.
4. In the **Tests** > **Test cases** tab, select a test case and click **Run**.



5. To run multiple test cases, select test cases with the status as **Published**, and click **Run**.

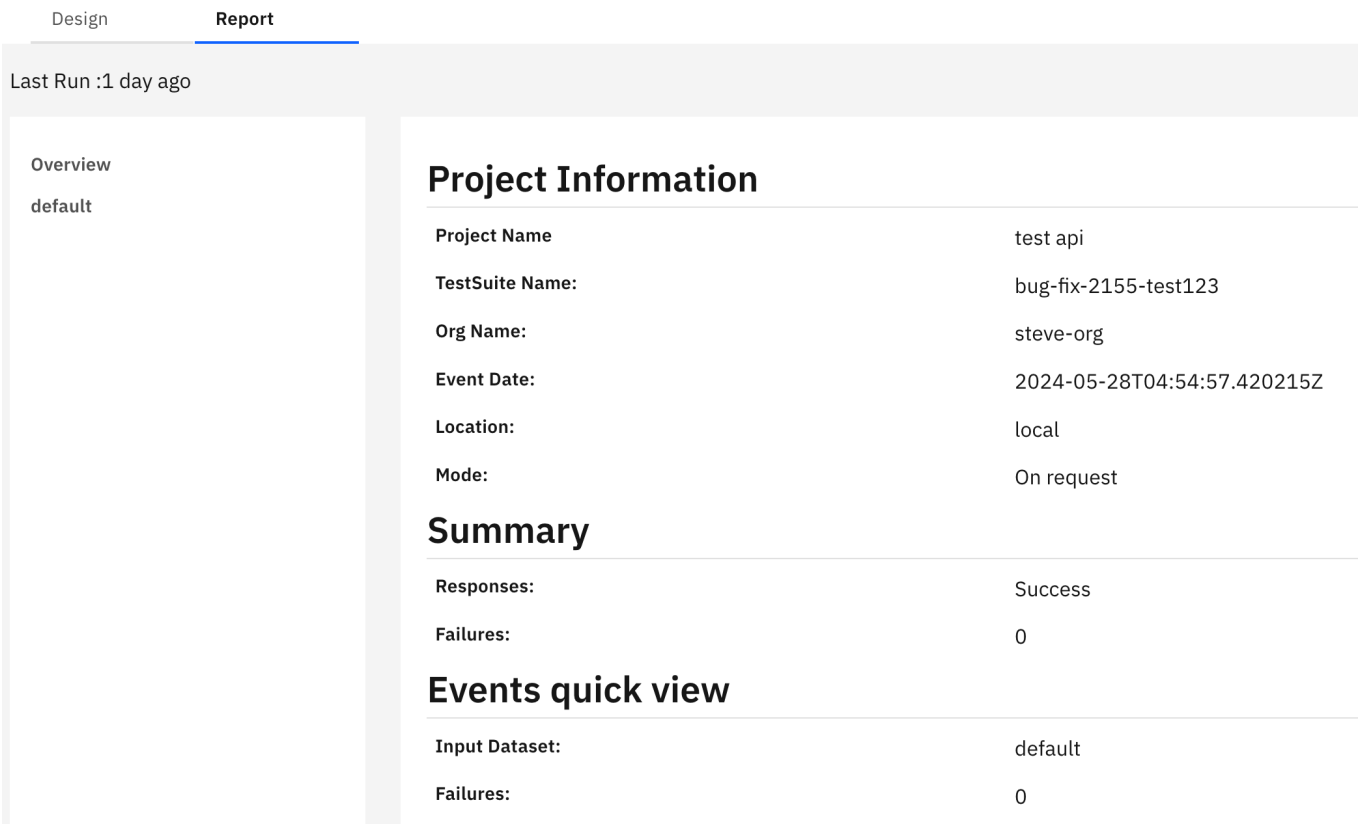
What to do next

- Next topic: Viewing a test report

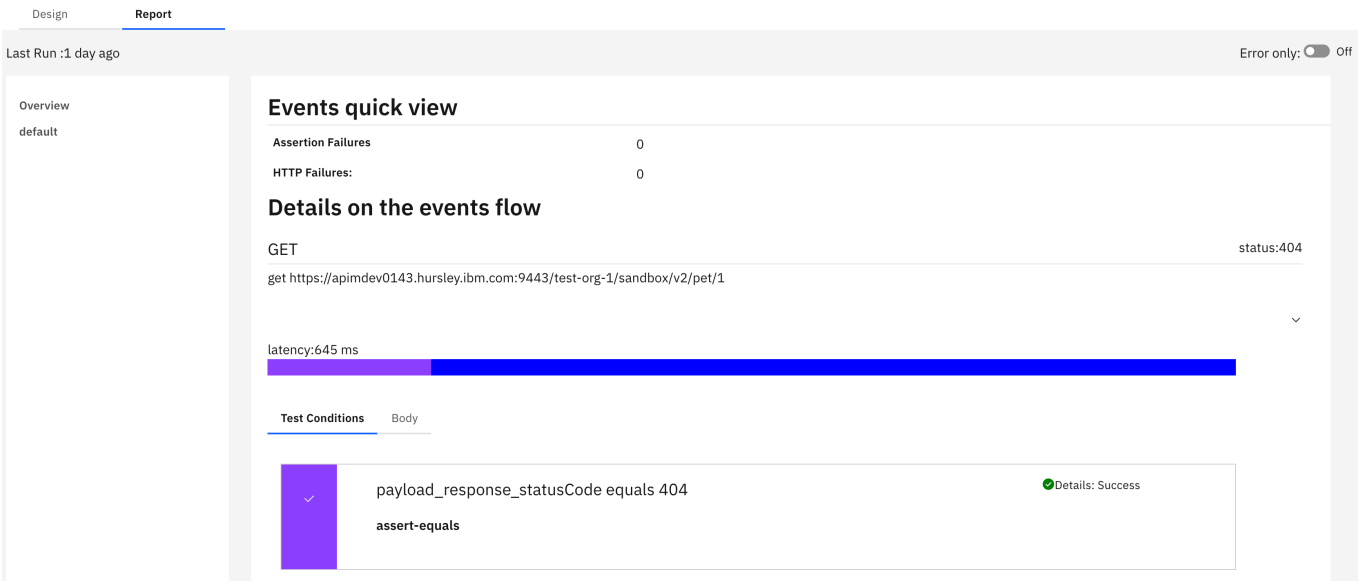
Viewing a test report

Once a test run has completed (either stopped manually or the configured time to run threshold has been reached) you will be presented with the **Report** section, where you will be able to view the results of the run.

First up is the **Overview** tab, which contains information about the test case. The project Information section details the test case, while the Summary section gives the overall test run status, indicating success or failure and the number of failures, if any. The Events quick view section offers insights into the failures associated with each input dataset



The second tab presents detailed results for each input dataset offering a comprehensive view of the test outcomes for each specific input. The table contains the list of the calls and their respective response statuses.



*Both views allow you to apply **Errors Only** filter using the toggle button. This filter displays only the failed test conditions.*

To download the report, click the options menu and select **Download Report**.

What to do next

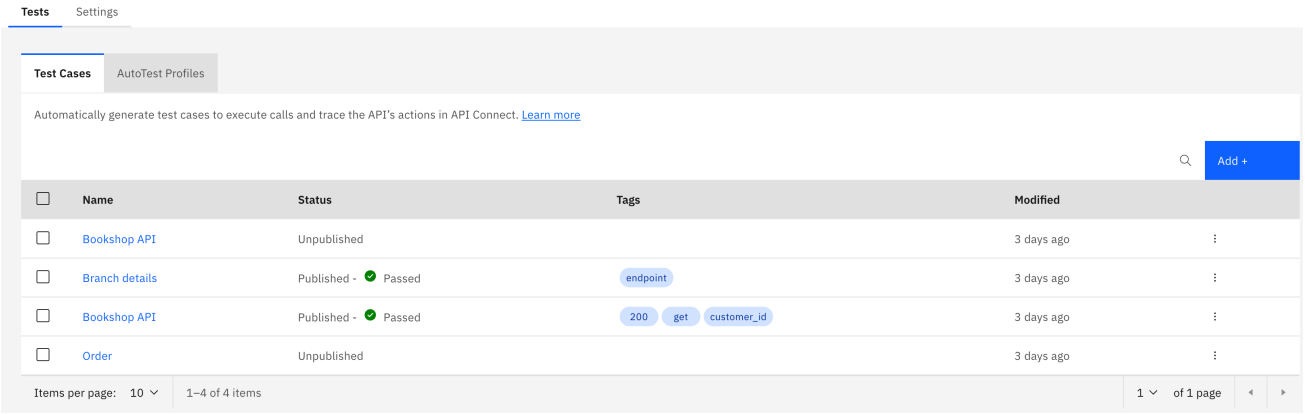
- Next topic: Publishing a test

Publishing a test case

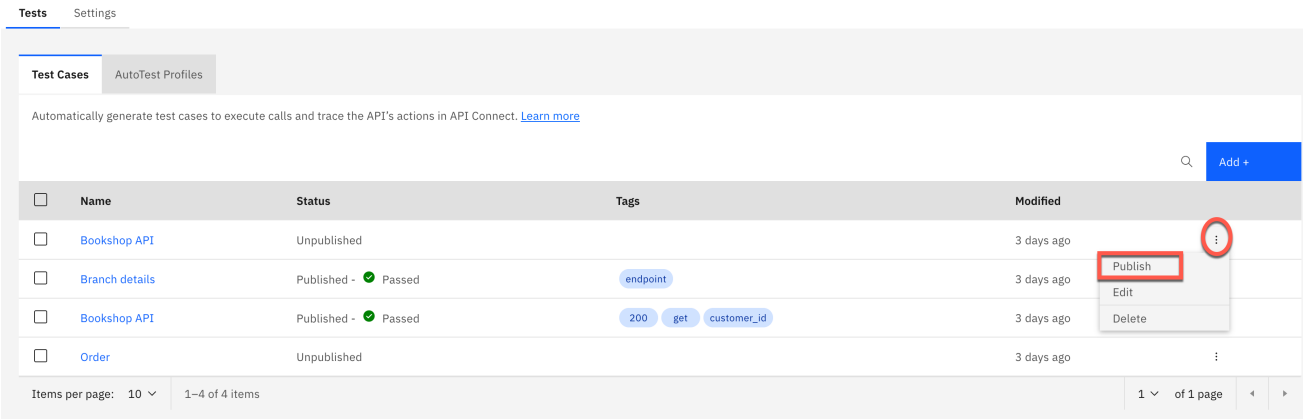
Once you have completed the configuration of your test, and run it manually to confirm that it works correctly, you can publish the test. After publishing, you have the option to select the published test case from the table, make changes using the editor, and publish it again.

To publish a test, complete the following steps:

1. Log in to the API Manager.
2. On the Home page, click the **Test APIs** tile.
3. On the Test page, select a test suite. A list of tests are displayed. If your test has not been published before, the status shows as **Unpublished**.



4. Select a test that you want to publish by clicking the **Options** icon and selecting **Publish**.



What to do next

- Next topic: Tagging a test

Tagging a test

Adding tags to your tests is an easy way to keep them organized in your Test Suite. Furthermore, tags can be used as a way to selectively execute certain tests in your test suite when running them via an API Hook (see Run Published Tests by Tag for more information).

Note: You can tag a test, while creating a test or after creating a test.

Adding a tag to a test

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite tile that contains the test that you want to tag.
4. Click the test case you want to edit.
5. To add tags to your test, go to the **Design > Info** tab and start typing the tag name in the Tags box. You can assign multiple tags to a test. To remove a tag, click the **x** button next to the tag.

6. Click **Save**. Tags can be viewed any time in the **Tests > Test Cases** tab.

What to do next

- Next topic: API Hooks

API Hooks

In addition to viewing and executing tests within the IBM API Connect Test interface, you may also perform these actions via an API. This allows tests to be executed via the command line, through third-party applications, or as part of a CI/CD pipeline. API hooks form the basis of this API, providing a base URL on which subsequent API operations can be performed. To start using an API Hook, see:

- Creating An API Hook.

API Hooks require authentication to use in the form of an API Key and Secret. Once you have created your API Hook, view the following page to learn how to create an API Key and Secret for your Test Organization:

- Creating API Keys and Secrets

Once you've set up your API Hook, Key and Secret, see:

- Using API Hooks

Creating an API Hook

You can generate an API hook for each of the Test Suites within your Test Organization. API hooks are scoped to only one test suite and thus will only have access to operate on the data within that test suite; that is, if an API Hook is generated for **`Test Suite 1`**, it may only access the tests and test results associated with **`Test Suite 1`**. If you wish to interact with tests in multiple test suites, you will need to generate an API hook for each Test Suite.

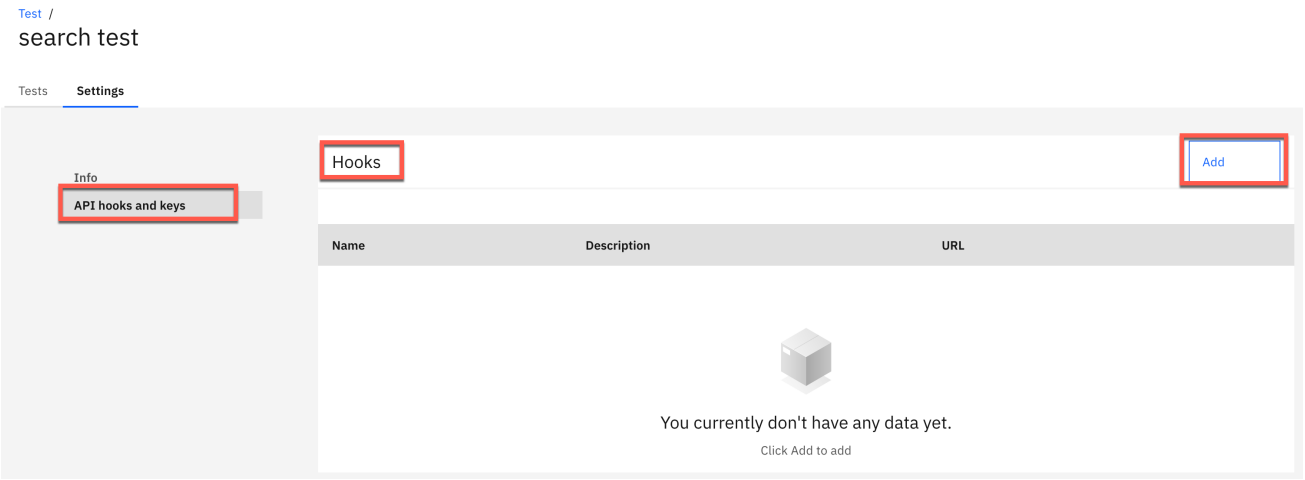
API Hooks are protected and require an API Key and Secret to be used. You can find out how to create a API Key and Secret here.

If there are no Test Suites here, you can create one by clicking the **Create Test Suite** button after the Test Status chart.

How to create an API Hook

1. Log in to the API Manager.
2. Click the Test APIs tile.
3. Select a test suite.

4. Select the **Settings** > **API hooks and keys** tab.



- Go to the Hooks section, and click **Add**.
- Enter name.
- Enter description.
- Click **Save**. You've now created the API Hook URL for your Test Suite. Make a note of this URL as this will form the base path of all future API requests you make to to this test suite. Close the Hook URL dialog box. You can see that your new API Hook is now in the list of API Hooks. You can click on this at anytime to view the URL again in the future should you need. Note that this URL will remain active until it is deleted. To delete the API Hook, simply click on the **Delete** icon next to the API Hook you want to remove.

What to do next

- Next Topic: Creating API Keys and Secrets

Creating API Keys and Secrets

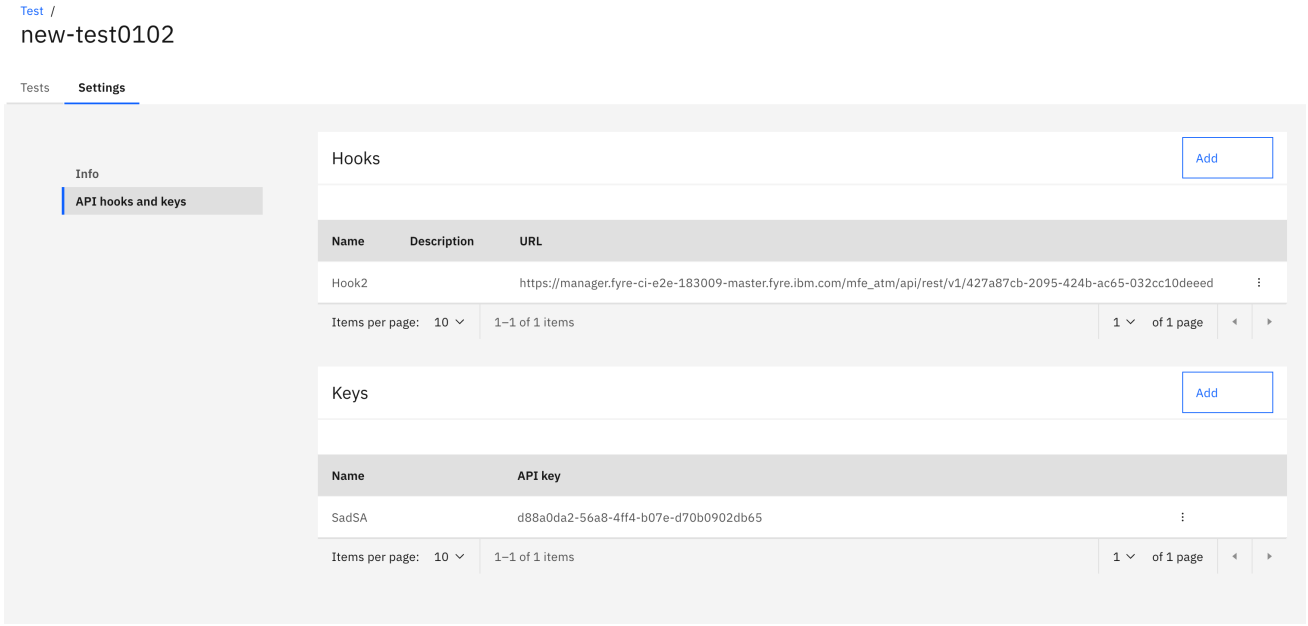
API Keys and Secrets are the credentials required to use an API Hook. For a request to an API Hook to be authorised, both the **`X-API-Key`** and **`X-API-Secret`** headers must be provided. The values of the API Key and Secret represent the values of these headers respectively.

Unlike API Hooks which are scoped to a single Test Suite, API Keys and Secrets are scoped to a Test Organization. Therefore, any API Key and Secret pair which is generated within a Test Organization will have permission to authorise requests to all API Hooks within the same organization, regardless of the Test Suite the API Hook is associated with.

How to create an API Key and Secret

1. Log in to the API Manager.
2. Click the Test APIs tile.
3. Select a test suite.
4. Go to the **Settings** > **API hooks and keys** tab.

5. To use your new API Hook, you need to ensure you have an API Key and Secret.



   - Go to the Keys section and click **Add**.
   - Enter the key name.
   - Click **Save**. The system will display the key and the secret. Copy it for future reference. if your Key and Secret is set up, see Using API Hooks for more information on how to use the API Hook.

Your new API Key is now active and will now appear in the list of API Keys in the navigation menu. Clicking on the key will show you the API Key's value should you need to reference this again in the future. If you want to remove the API Key at any point, simply click on the **Delete** icon next to the API Key you want to remove. This will invalidate all future requests to API Hooks using this Key and Secret combination.

What to do next

- Next Topic: Using API Hooks

Using API Hooks

API Hooks provide a mechanism to interact with tests within a Test Suite via an API. Whilst the API Hook itself defines the basepath of this API, it also exposes a series of endpoints which, when appended onto this basepath, allow you to list and execute published tests within the API Hook's scope. The following pages detail these endpoints and subsequent parameters you can provide to either list, execute all, or execute only specific tests within a test suite.

Endpoints

- List All Tests
  Lists all of the published tests within an API Hook's scope.
- Run All Published Tests
  Runs all of the published tests within an API Hook's scope.
- Run Published Tests By Tag
  Runs the published tests within an API Hook's scope that match the tags specified in the request body.
- Run Published Test By ID
  Run a specified test within an API Hook's scope.

List All Tests

Description

The List All Tests endpoint lists all of the published tests within an API Hook's scope.

HTTP Request

```
GET <API-Hook-URL>/tests
```

Sample Method Invocation

```
curl -X GET \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
<API-Hook-URL>/tests
```

Parameters	**Header Parameters** API-Key: A valid API Key for the Test Organization this API Hook belongs to. API-Secret: A valid API Secret for the Test Organization this API Hook belongs to.

Response Content	**MIME type: application/json**
	
	`[ { testId: string, testName: string, testDescription: string, lastModified: string, authorName: string, tags: array[string] } ]`
	
	Response Description
	• `testId`
	  ◦ ID of the test.
	• `testName`
	  ◦ Name of the test.
	• `testDescription`
	  ◦ Description of the test.
	• `lastModified`
	  ◦ Date and time the test was last modified.
	• `authorName`
	  ◦ User who composed the test.
	• `tags`
	  ◦ Array of tags associated with the test.

Status Codes	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers

Run All Published Tests

Description

The Run All Tests endpoint executes all of the published tests within an API Hook's scope.

HTTP Request

```
POST <API-Hook-URL>/tests/run
```

Sample Method Invocation

```
curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
```

```
    -H Content-Type:application/json \
    -d "
        {
          "options": {
              "allAssertions": boolean,
              "JUnitFormat": boolean
          },
          "variables": {
              string: string,
          }
        }
      " \
    <API-Hook-URL>/tests/run
```

Parameters	Header Parameters API-Key: A valid API Key for the Test Organization this API Hook belongs to. API-Secret: A valid API Secret for the Test Organization this API Hook belongs to.

Request Body	**MIME type: application/json**
	
	`{ options: { allAssertions: boolean, JUnitFormat: boolean }, variables: { string: string, } }`
	
	Body Parameters
	• `options` (Only required if providing a child element)
	  ◦ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.
	  ◦ Child Elements:
	  ▪ `allAssertions` (Optional)
	  ▪ When `false`, only the results of failing assertions in the test are returned. When `true`, all assertions in the test are returned. `allAssertions` is `false` by default.
	  ▪ `JUnitFormat` (Optional)
	  ▪ When `false`, `application/json` format is returned for the test results. When `true`, `text/xml` JUnit format is used for the test results (see `text/xml` response). `JUnitFormat` is `false` by default.
	• `variables` (Optional)
	  ◦ The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.

Response Content	**MIME type: application/json** `application/json` is the default response from the endpoint. `[{ testRunId: string, testId: string, testName: string, location: string, initiated: string, duration: integer, status: enum['passed' or 'failed'], reportUrl: string, results: { warningsCount: integer, failureCount: integer, httpFailures: array[object], criticalFailures: array[object], assertions: array[object] } }]` **Response Description**

- `testRunID`
  - ID of the test run.
- `testId`
  - ID of the test which was executed.
- `testName`
  - Name of the test.
- `location`
  - Location of the agent which excuted the test.
- `initiated`
  - Date and time when the test was initiated.
- `duration`
  - Total duration of the test run in seconds.
- `status`
  - Overall result of the test.
- `reportUrl`
  - URL to the test run report.
- `results`
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - `warningsCount`
      - Count of the HTTP warnings returned by the test assertions.
    - `failureCount`
      - Count of the HTTP status code responses that are greater than 400.
    - `httpFailures`
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - `criticalFailures`
      - Description of failures whose HTTP Status response code is greater than or equal to 500.
    - `assertions`
      - A description of the assertions executed in the test. By default, only failing assertions will be listed.

MIME type: text/xml

To receieve a `text/xml` response, the optional `JUnitFormat` variable in the request body must be set to `true`. JUnit responses can be useful when you wish to incorporate your test reults with a build pipline such as Jenkins, via the JUnit plugin.

```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
<testcase name="" classname="" time="">
  <failure message="" type="" />
</testcase>
</testSuite>
```

Response Description

- `<testSuite>`
  - Child Elements: `<testCase>`
  - Attributes:
    - `name`
      - Test Suite name.
    - `tests`
      - Number of published tests ran.
    - `disabled`
      - Number of disabled tests.
    - `errors`
      - Number of errors.
    - `failures`
      - Number of tests within the test suite that failed.
    - `timestamp`
      - Date and time the tests were ran.
    - `time`
      - Total duration of all test runs in seconds.
- `<testCase>`
  - Child Elements: `<failure>`
  - Attributes:
    - `name`
      - Test name.
    - `classname`
      - ID of the test which was executed.
    - `time`
      - Total duration of the test run in seconds.
- `<failure>`
  - Attributes:
    - `message`

	▪ Failure message containing the expected and actual response. ▪ `type` ▪ Type of failure.
Status Codes	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers

Run Published Tests By Tag

Description

The Run Published Test By Tag endpoint executes all of the published tests within an API Hook's scope that match the tags specified in the request body. See Tagging a Test for more information on how to tag a test.

HTTP Request

```
POST <API-Hook-URL>/tests/run/tag
```

Sample Method Invocation

```
curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d "
    {
      "options": {
          "allAssertions": boolean,
          "JUnitFormat": boolean
      },
      "variables": {
          string: string,
      },
      "tags": {
        "tagList": array[string],
        "operator": enum['not' or 'and' or 'or']
      }
    }
  " \
<API-Hook-URL>/tests/run/tag
```

Parameters	**Header Parameters** API-Key: A valid API Key for the Test Organization this API Hook belongs to. API-Secret: A valid API Secret for the Test Organization this API Hook belongs to.

Request Body	**MIME type: application/json**
	
	`{`
	`options: {`
	`allAssertions: boolean,`
	`JUnitFormat: boolean`
	`},`
	`variables: {`
	`string: string,`
	`},`
	`tags: {`
	`tagList: array[string],`
	`operator: enum['not' or 'and' or 'or']`
	`}`
	`}`
	
	Body Parameters
	
	• `options` (Only required if providing a child element)
	○ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.

- Child Elements:
  - **allAssertions** (Optional)
    - When **false**, only the results of failing assertions in the test are returned. When **true**, all assertions in the test are returned. **allAssertions** is **false** by default.
  - **JUnitFormat** (Optional)
    - When **false**, **application/json** format is returned for the test results. When **true**, **text/xml** JUnit format is used for the test results (see **text/xml** response). **JUnitFormat** is **false** by default.
- **variables** (Optional)
  - The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.
- **tags** (Required)
  - A wrapper object containing both the list of tags and the operator to use on them.
  - Child Elements:
    - **tagList** (Required)
      - The list of tags which correspond to the tests you want to execute or ignore (see **operator** element).
    - **operator** (Required)
      - The operator indicates how to treat the list of tags and can be either one of three values: **not**, **and**, **or**.
        - The **not** operator will omit executing any test within the API Hook's scope that has been tagged with a tag listed in the **tagList**.
        - The **and** operator will include any test within the API Hook's scope that has been tagged with all the tags provided in the **tagList**.
        - The **or** operator will include any test within the API Hook's scope that has been tagged with at least one of the tags provided in the **tagList**.

Response Content

MIME type: application/json

application/json is the default response from the endpoint.

```
[
{
  testRunId: string,
  testId: string,
  testName: string,
  location: string,
  initiated: string,
  duration: integer,
  status: enum['passed' or 'failed'],
  reportUrl: string,
  results: {
    warningsCount: integer,
    failureCount: integer,
    httpFailures: array[object],
    criticalFailures: array[object],
    assertions: array[object]
  }
}
]
```

Response Description

- **testRunID**
  - ID of the test run.
- **testId**
  - ID of the test which was executed.
- **testName**
  - Name of the test.
- **location**
  - Location of the agent which excuted the test.
- **initiated**
  - Date and time when the test was initiated.
- **duration**
  - Total duration of the test run in seconds.
- **status**
  - Overall result of the test.
- **reportUrl**
  - URL to the test run report.
- **results**
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - **warningsCount**
      - Count of the HTTP warnings returned by the test assertions.
    - **failureCount**
      - Count of the HTTP status code responses that are greater than 400.
    - **httpFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - **criticalFailures**
      - Description of failures whose HTTP Status response code is greater than or equal to 500.

- `assertions`
  - A description of the assertions executed in the test. By default, only failing assertions will be listed.

MIME type: text/xml

To receieve a `text/xml` response, the optional `JUnitFormat` variable in the request body must be set to `true`. JUnit responses can be useful when you wish to incorporate your test reults with a build pipline such as Jenkins, via the JUnit plugin.

```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
<testcase name="" classname="" time="">
  <failure message="" type="" />
</testcase>
</testSuite>
```

Response Description

- `<testSuite>`
  - Child Elements: `<testCase>`
  - Attributes:
    - `name`
      - Test Suite name.
    - `tests`
      - Number of published tests ran.
    - `disabled`
      - Number of disabled tests.
    - `errors`
      - Number of errors.
    - `failures`
      - Number of tests within the test suite that failed.
    - `timestamp`
      - Date and time the tests were ran.
    - `time`
      - Total duration of all test runs in seconds.
- `<testCase>`
  - Child Elements: `<failure>`
  - Attributes:
    - `name`
      - Test name.
    - `classname`
      - ID of the test which was executed.
    - `time`
      - Total duration of the test run in seconds.
- `<failure>`
  - Attributes:
    - `message`
      - Failure message containing the expected and actual response.
    - `type`
      - Type of failure.

Status Codes	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers

Run Published Test By ID

Description

The Run Published Test By ID endpoint executes a single specified test within an API Hook's scope.

HTTP Request

```
POST <API-Hook-URL>/tests/{Test-ID}/run
```

Sample Method Invocation

```
curl -X POST \
-H X-API-Key:{API-Key} \
-H X-API-Secret:{API-Secret} \
-H Content-Type:application/json \
-d "
    {
      "options": {
          "allAssertions": boolean,
          "JUnitFormat": boolean
      },
      "variables": {
          string: string,
      }
    }
  " \
<API-Hook-URL>/tests/{Test-ID}/run
```

Parameters	**Path Parameters** Test-ID: The ID of the Test to execute. **Header Parameters** API-Key: A valid API Key for the Test Organization this API Hook belongs to. API-Secret: A valid API Secret for the Test Organization this API Hook belongs to.

Request Body	**MIME type: application/json**
	
	`{ options: { allAssertions: boolean, JUnitFormat: boolean }, variables: { string: string, } }`
	
	Body Parameters
	
	• `options` (Only required if providing a child element)
	  ◦ A wrapper object for the containing options on how what data should be returned in the response, or how the response should be formatted.
	  ◦ Child Elements:
	  ▪ `allAssertions` (Optional)
	  ▪ When `false`, only the results of failing assertions in the test are returned. When `true`, all assertions in the test are returned. `allAssertions` is `false` by default.
	  ▪ `JUnitFormat` (Optional)
	  ▪ When `false`, `application/json` format is returned for the test results. When `true`, `text/xml` JUnit format is used for the test results (see `text/xml` response). `JUnitFormat` is `false` by default.
	• `variables` (Optional)
	  ◦ The variables section is used to pass a list of named variables and their values into the tests being run. It is only used by tests that have been written to make use of variables, and have values stored alongside them. Declaring the variable name as the object key, and its new value as the key's value (as type string), will overwrite the current value of the variable in the test which shares its name with the object key.

Response Content	**MIME type: application/json** `application/json` is the default response from the endpoint. `[{ testRunId: string, testId: string, testName: string, location: string, initiated: string, duration: integer, status: enum['passed' or 'failed'], reportUrl: string, results: { warningsCount: integer, failureCount: integer, httpFailures: array[object], criticalFailures: array[object], assertions: array[object] }`

```
    }
    ]
```

Response Description

- `testRunID`
  - ID of the test run.
- `testId`
  - ID of the test which was executed.
- `testName`
  - Name of the test.
- `location`
  - Location of the agent which excuted the test.
- `initiated`
  - Date and time when the test was initiated.
- `duration`
  - Total duration of the test run in seconds.
- `status`
  - Overall result of the test.
- `reportUrl`
  - URL to the test run report.
- `results`
  - A wrapper object containing the results of the test run.
  - Child Elements:
    - `warningsCount`
      - Count of the HTTP warnings returned by the test assertions.
    - `failureCount`
      - Count of the HTTP status code responses that are greater than 400.
    - `httpFailures`
      - Description of failures whose HTTP Status response code is greater than or equal to 400.
    - `criticalFailures`
      - Description of failures whose HTTP Status response code is greater than or equal to 500.
    - `assertions`
      - A description of the assertions executed in the test. By default, only failing assertions will be listed.

MIME type: text/xml

To receieve a `text/xml` response, the optional `JUnitFormat` variable in the request body must be set to `true`. JUnit responses can be useful when you wish to incorporate your test reults with a build pipline such as Jenkins, via the JUnit plugin.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <testSuite name="" tests="" disabled="" errors="" failures="" timestamp="" time="">
    <testcase name="" classname="" time="">
      <failure message="" type="" />
    </testcase>
    </testSuite>
```

Response Description

- `<testSuite>`
  - Child Elements: `<testCase>`
  - Attributes:
    - `name`
      - Test Suite name.
    - `tests`
      - Number of published tests ran.
    - `disabled`
      - Number of disabled tests.
    - `errors`
      - Number of errors.
    - `failures`
      - Number of tests within the test suite that failed.
    - `timestamp`
      - Date and time the tests were ran.
    - `time`
      - Total duration of all test runs in seconds.
- `<testCase>`
  - Child Elements: `<failure>`
  - Attributes:
    - `name`

	▪ Test name. ▪ `classname` ▪ ID of the test which was executed. ▪ `time` ▪ Total duration of the test run in seconds. • `<failure>` ◦ Attributes: ▪ `message` ▪ Failure message containing the expected and actual response. ▪ `type` ▪ Type of failure.
Status Codes	200: Success 400: Missing Authorization Headers 401: Invalid Authorization Headers

Test definition syntax

The test is defined as a YAML file with the following sections:

```
- type: request
  method: get
  url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/bank/branches
  var: branches
  mode: json
- type: assert-is
  expression: branches
  value: array
- type: assert-exists
  expression: branches.[1]
  stoponfail: true
- type: assert-is
  expression: branches.[0].type
  value: string
  stoponfail: true
- type: assert-is
  expression: branches.[0].address.city
  value: string
  stoponfail: true
- type: assert-compares
  expression1: branches.[0].type
  expression2: branches.[7].type
- type: request
  method: get
  url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/orders/order/AB1234
  var: payload
  mode: json
- type: assert-equals
  expression: payload_response_statusCode
  value: 200
  ifexists: false
  stoponfail: false
- type: assert-equals
  expression: payload_response_header_Content-Type
  value: application/json
- type: assert-in
  expression: payload_response_header_Content-Type
  values:
    - application/json
    - application/swagger+yaml
- type: assert-contains
  expression: payload_response_header_Content-Type
  value: app
- type: assert-matches
  expression: payload_response_header_Content-Type
  value: ^[a-z]+/json
- type: assert-exists
  expression: "payload"
  stoponfail: false
- type: assert-is
  expression: "payload.tracking_reference"
  value: "string"
  ifexists: "false"
  stoponfail: false
- type: assert-exists
  expression: payload.shipped_at
  stoponfail: false
- type: assert-exists
  expression: payload.status
  stoponfail: false
- type: assert-exists
  expression: payload.created_at
  stoponfail: false
```

Table 1. Summary of the parameters supported by the type attribute

Parameters	Description	Sub-attributes
request	Make an HTTP request.	- method: Defines the HTTP method to use. - url: Defines the URL to make the request to. - params: Defines the key value mapping of parameters to send. - headers: Defines the key value mapping of headers to send. - var: Defines the name of the variable to store the response in. - mode: Defines what type of data to expect and how to parse it (supported: json). For example: - type: request - method: get - url: https://sample-api.us-east-a.apiconnect.automation.ibm.com/bank/branches - var: branches - mode: json
assert-is	Assert that the type of variable from the response is value.	- expression: Defines the variable to test. - value: Defines the type we expect in response. For example: - type: assert-is expression: branches value: array
assert-exists	Assert that a particular variable specified in expression exists.	- expression: Defines the variable to test. - stoponfail: If one of the tests fails, the stoponfail attribute stops the execution of the remaining tests. For this, stoponfail must be set to true. For example: - type: assert-exists expression: branches.[1] stoponfail: true
assert-equals	Assert that a variable is equal to value.	- expression: Defines the variable to test. - value: Defines the value to compare against. For example: - type: assert-equals expression: payload_response_header_Content-Type value: application/json
assert-greater	Assert that the variable is greater than value.	- expression: Defines the variable to test. - value: Defines the value to compare against. For example: - type: assert-greater expression: payload_temperature value: application/json
assert-less	Assert that the variable is lesser than value.	- expression: Defines the variable to test. - value: Defines the value to compare against. For example: - type: assert-less expression: payload_temperature value: application/json
assert-in	Assert that the element identified by a given expression matches at least one item from a given list.	- expression: Defines the variable to test. - value: Defines an array of values to compare against. For example: - type: assert-in expression: payload_response_status value: application/json
assert-matches	Assert that the value of the element identified by a given expression matches a specified format.	- expression: Defines the variable to test. - value: Defines the regex value to compare against. For example: - type: assert-matches expression: payload_response_header_Content-Type value: [1]+/json
assert-compares	Assert that two elements are equivalent in some way.	- expression1: Defines the path to the first element for comparison. - expression2: Defines the path to the second element for comparison. For example: - type: assert-compares expression1: branches.[0].type expression2: branches.[7].type
assert-contains	Assert that the value of the element identified by a given expression contains a specific substring.	- expression: Defines the variable to test. - value: Defines the substring to compare against. For example: - type: assert-contains expression: payload_response_header_Content-Type value: app

Parameters	Description	Sub-attributes
each	Loop over the array referenced in the expression carrying out the sub-steps for each item found.	- expression: Defines the variable to test. For example: - type: each expression: "analytics.events"

1. a-z ↩

Tutorial: Invoking test cases from Jenkins with JUnit result format

Before you begin

You must have the Automated API behavior testing application enabled and have created and published at least one test. You must have a Jenkins instance that you can create a new build in with the JUnit Plugin installed

About this tutorial

In this tutorial, you invoke a test case using injected variables from a jenkins pipeline

This tutorial takes you through the following steps

1. Creating an API Hook
2. Creating an API Key and Secret
3. Creating a Jenkins build that invokes the API Hook
4. Executing the build and reviewing the test results

Creating an API Hook

See Creating an API Hook

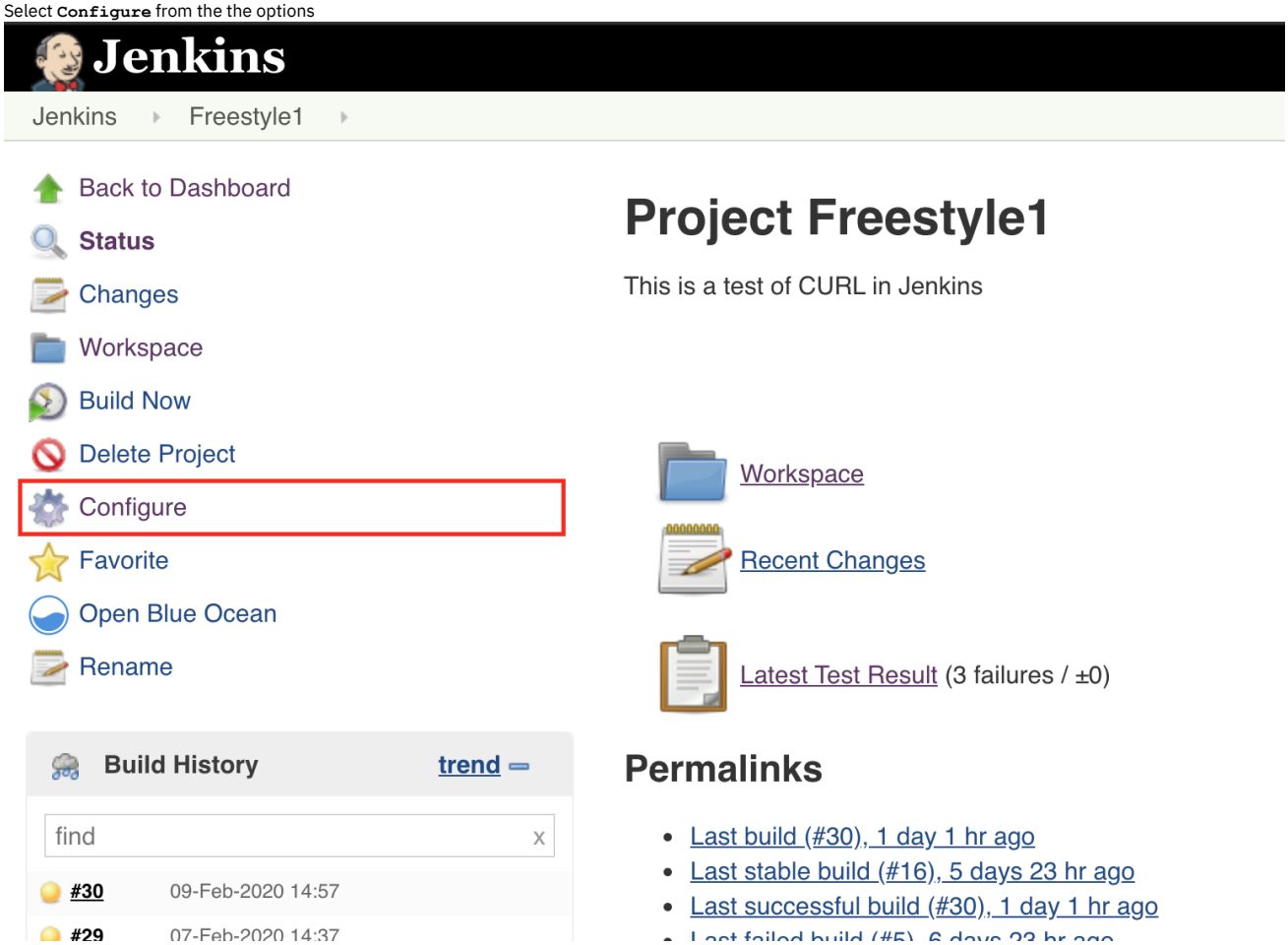
Creating an API Key and Secret

See Creating an API Key and Secret

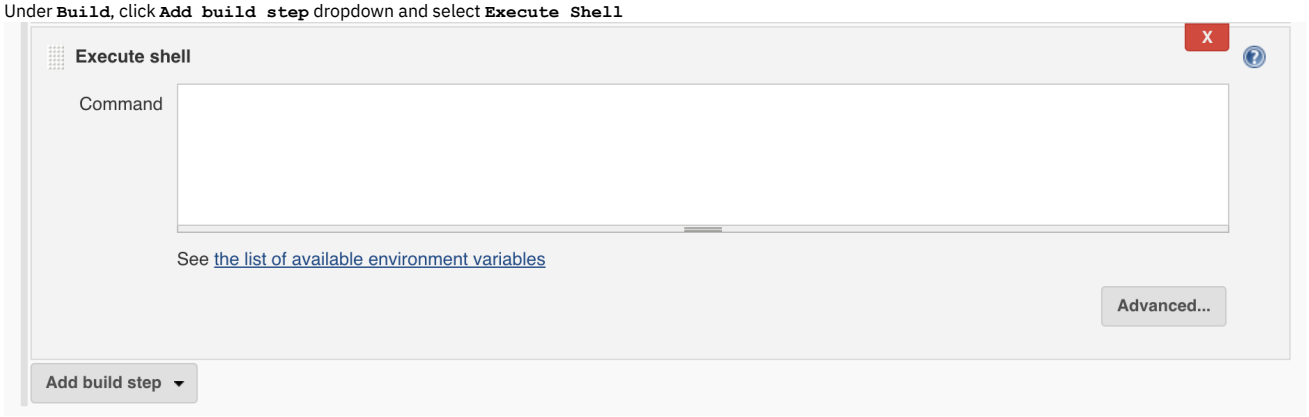
Creating a Jenkins build that invokes the API Hook

1. Create a Jenkins Project or navigate to an existing one

2. Select **`Configure`** from the the options



3. Under **`Build`**, click **`Add build step`** dropdown and select **`Execute Shell`**

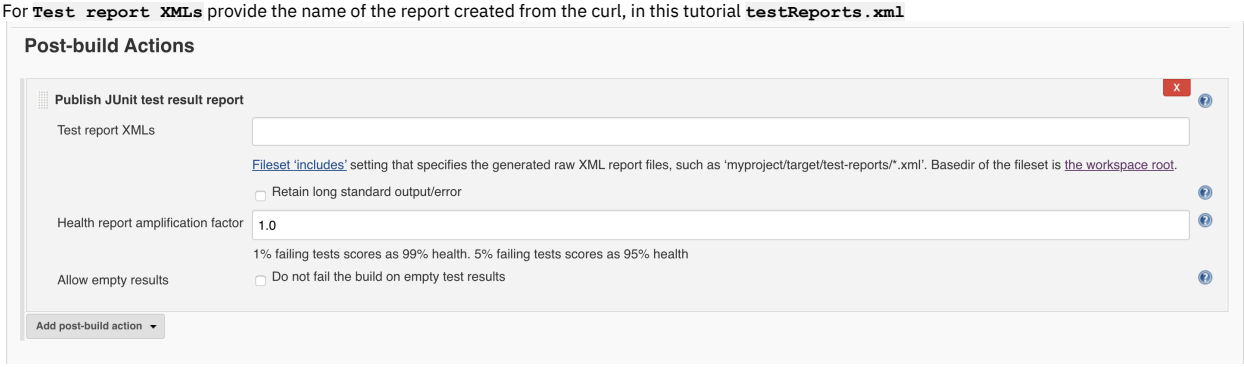


4. Within the **`Execute shell`** block paste a **`curl`** command that will invoke the hook that was created in step 1

```
curl -XPOST
-H 'x-api-key: <API-KEY>'
-H 'x-api-secret: <API-SECRET>'
-H "Content-type: application/json"
-d '{ options: { JUnitFormat: true }}'
'<API-HOOK-URL>/tests/run'
>| testReports.xml
```

5. Under **`Post-build Actions`**, click **`Add post-build action`** and in the dropdown select **`Publish JUnit test results report`**.

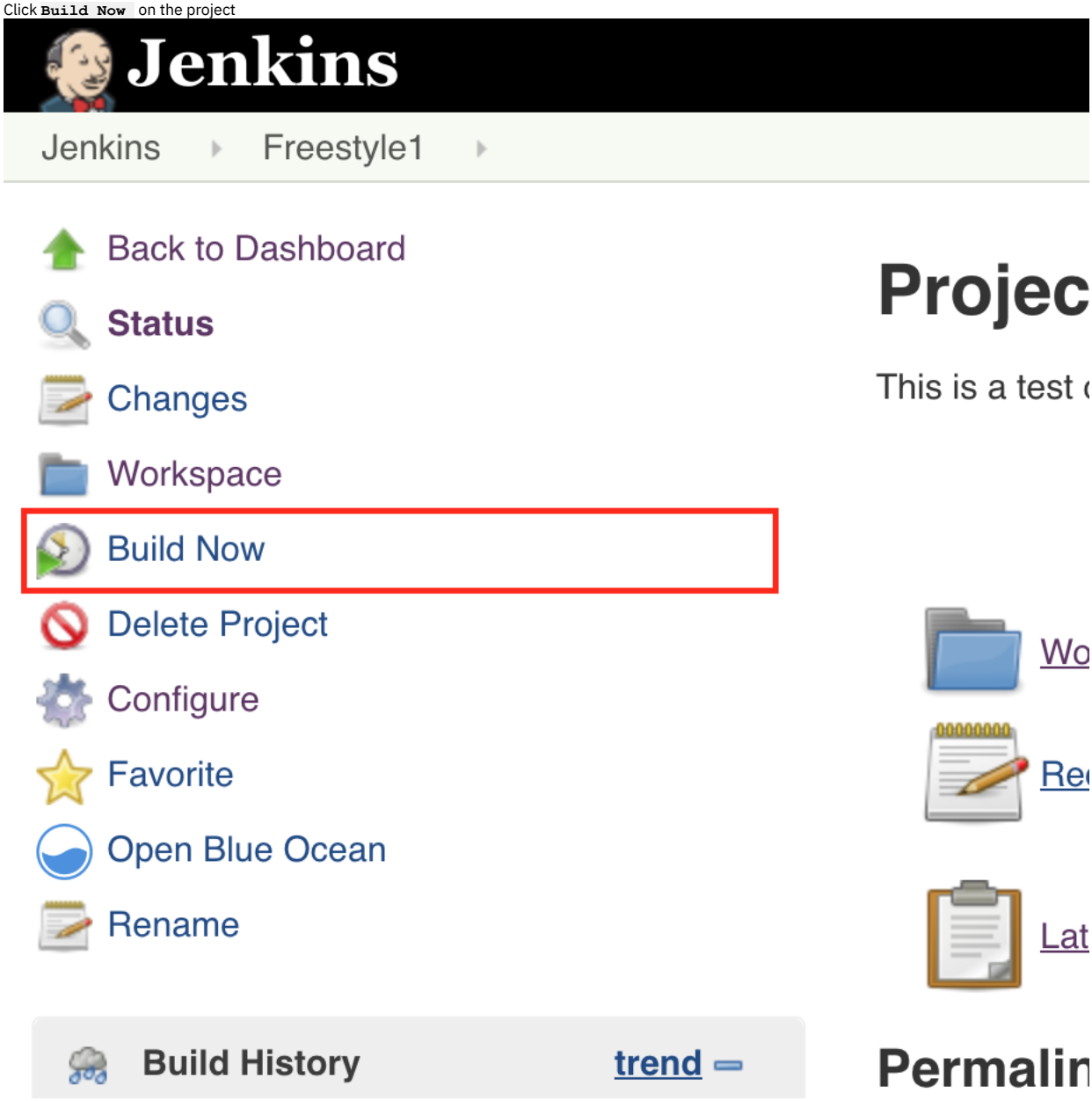
1. For `Test report XMLs` provide the name of the report created from the curl, in this tutorial `testReports.xml`



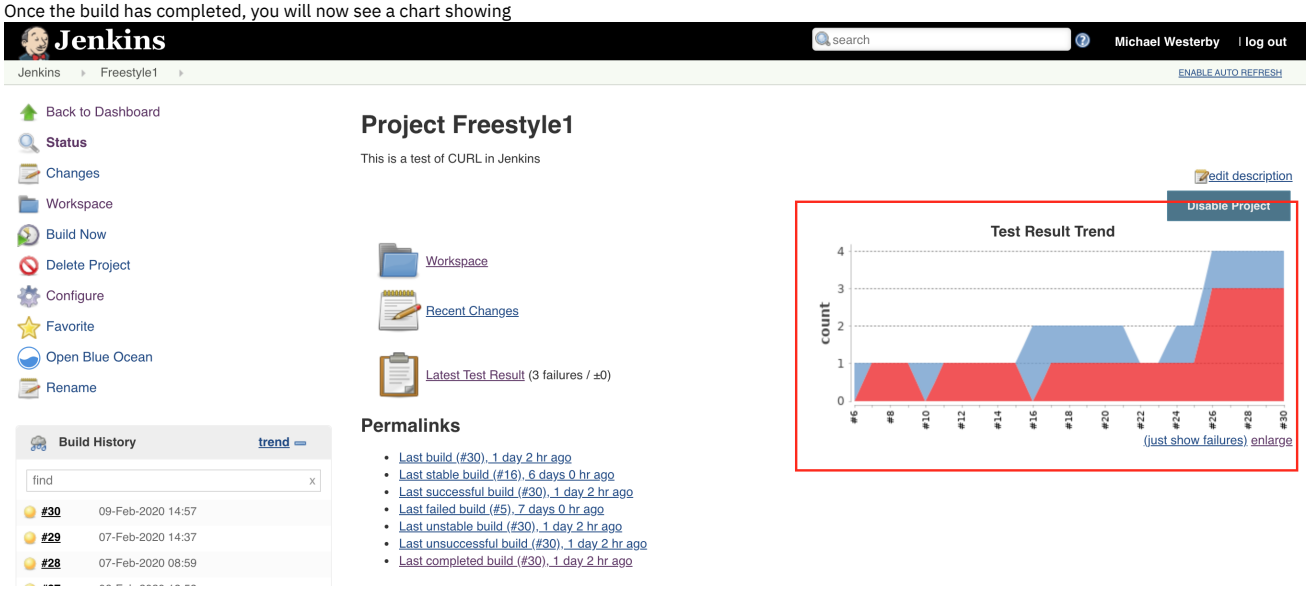
6. Save the project

Executing the build and reviewing the test results

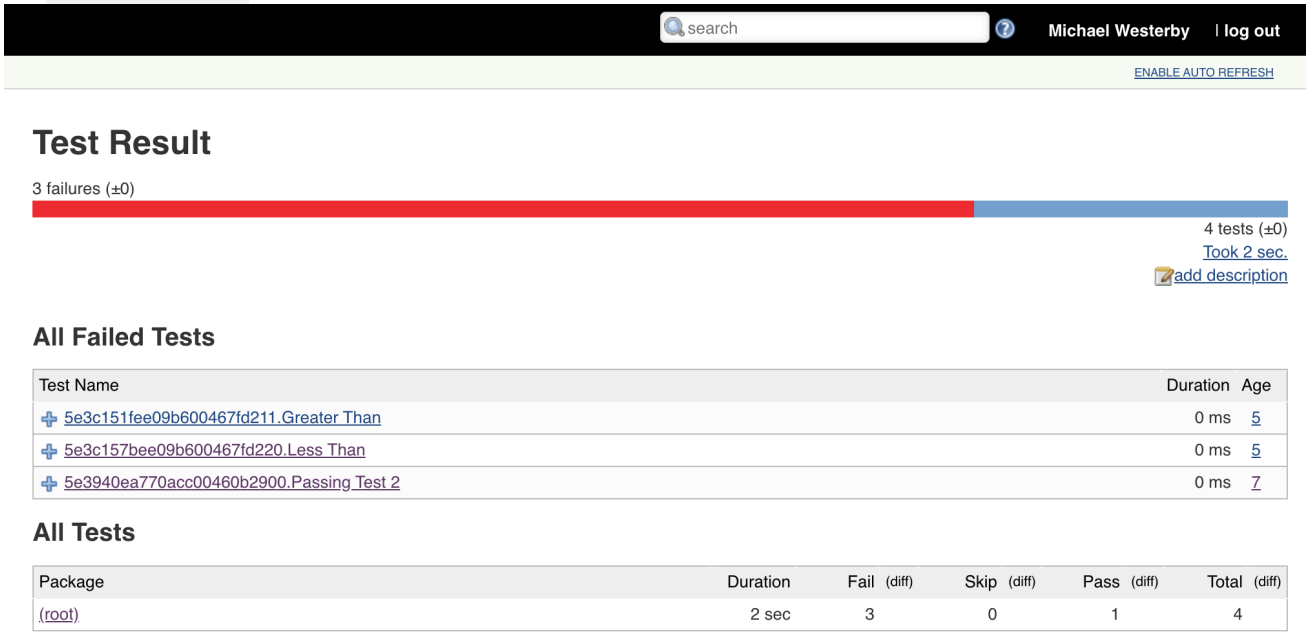
1. Click `Build Now` on the project



2. Once the build has completed, you will now see a chart showing



3. Click **`Latest Test results`** and review the Junit results



To expand upon this tutorial additional options and steps can be found under Using API Hooks

What you did in this tutorial

In this tutorial you completed the following activities:

- Created a API Hook
- Created a Key and Secret used for authenicating with a hook
- Invoked the API Hook using the Key and Secret to start a test and process the results

AutoTest Assist

This feature is available only in API Connect Enterprise as a Service

The AutoTest assistant exercises your API by sending it a series of randomly-generated requests, guided by the API definition and any custom rules that you provide, and alerts you about any conformance errors and server-side errors that it observes.

You control the auto-tester by creating and running an AutoTest *profile,* and you can create multiple profiles against the same API to cover different use-cases. A profile specifies how the auto-tester should run, which endpoints it should test, and what kind of data it should send. A default profile is constructed so that you can get immediate feedback by providing no more than a target URL and credentials.

The AutoTest assistant is especially useful while an API is under development as it can quickly identify errors in an implementation without you having to write test cases. It is also valuable later in the cycle for spotting regressions, as the random nature of its requests can provide coverage that your existing test cases do not match.

However, it is not meant as a replacement for other testing tools in API Connect, but is rather complementary, and the information it provides when it identifies an error is often a good starting point for creating a test case.

Follow these steps to create and run your first profile:

- Creating a profile
- Modifying a profile
- Deleting a profile
- Running a profile
- Viewing the report

Consult these additional guides for detailed instructions on how to configure the auto-tester to support your use case:

- Profile Configuration
- Profile Security
- Data Generation Rules

Known limitations

- The API definition must conform to OpenAPI 2.0 (Swagger 2.0)
- The API definition document must have a `.json` or `.yaml` extension (you cannot select a document with another extension)
- If there are errors in the API definition then you will see only a blank screen, and the errors are not reported
- You must provide either a username and password, or a bearer token, to authenticate against the target API; other authentication methods are not supported
- You can run only one profile at a time

Creating an AutoTest profile

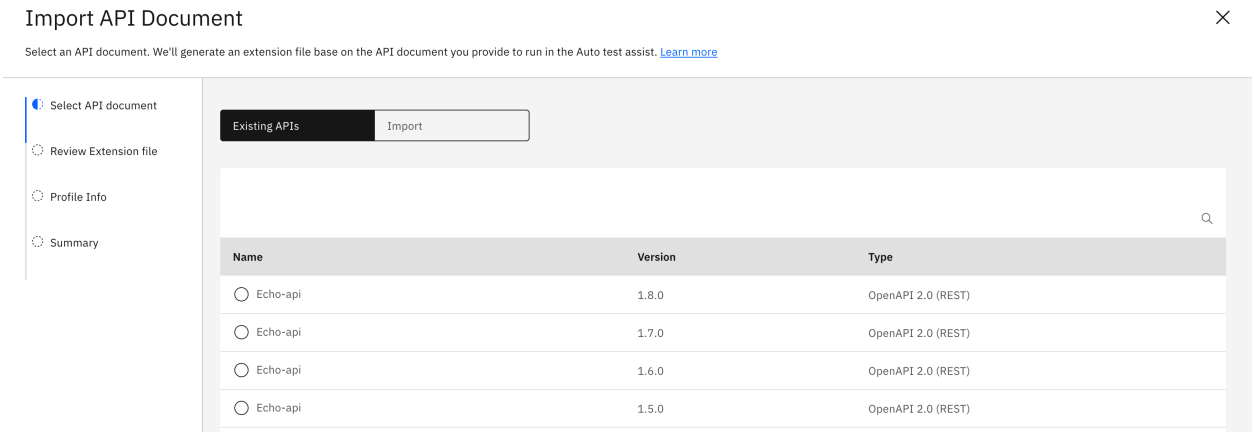
An AutoTest profile exists within a test suite, alongside your test cases.

To add an AutoTest profile to a test suite:

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite.
4. Go to the **Tests** > **AutoTest Profiles** tab.
5. Click **Add**.
6. Create an AutoTest profile using one of the following methods:

Existing APIs

1. Select a predefined API.



2. Click **Next**.
3. Review the read-only extension file.
4. Enter the profile name, description, and tags.
5. Click **Next**. You will see the message that the extension file and the AutoTest configuration has been generated.

Import

1. Drag and drop a JSON or YAML file that contains the API test case definition, or enter a URL for the API test case and provide a username and password if required for authentication.
2. Click **Next**.
3. Review the extension file and click **Next**.
4. Enter the profile name, description, and tags.
5. Click **Next**. You will see the message that the extension file and the AutoTest configuration has been generated.

What to do next

- Next topic: Running an AutoTest profile

Modifying an AutoTest profile

To edit an AutoTest profile in a test suite:

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite.
4. Go to the **Tests** > **AutoTest Profiles** tab.
5. Select an AutoTest profile.
6. Go to the **Info** tab, update the profile name, description, and tags, then click **Save**.
7. To edit the extension file, click the extension file, update the settings, and click **Save**. For more information, see API Extensions

What to do next

- Next topic: Running an AutoTest profile

API Extensions

An **API Extension** augments an OpenAPI definition with semantic information designed to improve the automatic generation of test cases. Extensions are used primarily by the AutoTest assistant to refine its randomly-generated test cases, but have wider applicability.

An extension comes in two parts:

- Resources identify the key resource types managed by the API, and their methods
- Properties attach semantic categories to schema properties, to constrain the format of values suggested for those fields

An extension is associated with a specific OpenAPI document and uses JSON Pointer notation to locate terms in that document.

Resources

The **`resources`** section contains a set of resource definitions, where each definition contains at least the following:

- a name for the resource
- a pointer to a schema of type **`object`** that models the resource
- a collection of methods that operate on the resource

Typically, a resource definition also identifies a field within the schema that stores a unique identifier for instances of that type.

For example, consider this extract from an API extension:

```
resources:
  Book:
    schemas:
      primary:
        json_ptr: '#/definitions/Book'
    properties:
      id_name: '$.book_id'
    operations:
      create:
        - json_ptr: '#/paths/~1books/post'
      retrieve:
        - json_ptr: '#/paths/~1books~1{book_id}/get'
        - json_ptr: '#/paths/~1books/get'
      update:
        - json_ptr: '#/paths/~1books~1{book_id}/put'
      delete:
        - json_ptr: '#/paths/~1books~1{book_id}/delete'
```

The **`resources`** section contains a single resource definition with the following properties:

- it defines a resource named **`Book`**
- a book is modeled by a schema of the same name located in the OpenAPI document under **`definitions`**
- each book has a property **`book_id`** that holds its unique ID
- there are operations defined in the OpenAPI document under **`paths`** that create, retrieve, update and delete books

The correspondence between the resource name and the schema name is common but not required: a resource name can be anything, as long as it is unique within the API.

The unique ID field for a book is identified by a simplified form of JSON Path expression that locates a property (**`book_id`**) within the primary schema.

Operations (or methods) on a resource are grouped by category:

- Create
- Retrieve
- Update
- Delete

There can be multiple operations in each. In this example there are two operations that retrieve books: one that returns a single book by ID, and another that returns a list of books, and this is common. A different definition might have **`patch`** as well as (or instead of) **`put`** under **`update`**.

JSON Pointer expressions in a resource definition all refer implicitly to the associated OpenAPI document, so the prefix **`#`** introduces a relative location within that document. JSON Pointer syntax uses the string **`~1`** to stand for a forward slash character (**`/`**) to prevent confusion with the same character used as a separator, so the operation defined in the OpenAPI document by the term:

```
paths:
  /books/{book_id}:
    get:
```

is identified here by the JSON Pointer:

```
#/paths/~1books~1{book_id}/get
```

Dependencies

A resource definition may also describe a dependency from one resource type to another.

For example:

```
  Order:
    schemas:
      primary:
        json_ptr: '#/definitions/Order'
    properties:
      id_name: '$.order_id'
    dependencies:
      - name: Book
        required: true
        references:
          - name: $.book_id
            in: body
        dependee_deletion: disabled
      - name: Customer
        required: true
        references:
          - name: customer_id
            in: path
        dependee_deletion: mutual
    operations:
      create:
        - json_ptr: '#/paths/~1customers~1{customer_id}~1orders~1{order_id}/post'
      # rest omitted for brevity
```

This definition says that the resource **`Order`** has dependencies on two other resources **`Customer`** and **`Book`** (which must be defined in the same extension).

Both dependencies are **required** in the sense that an order cannot exist unless there is a customer (to place the order) and at least one book (to buy).

An order refers to an existing book through the property **`book_id`** in the primary schema, and to a customer through the path parameter **`customer_id`**, as in the **`create`** method shown (and in other methods omitted).

A customer *owns* a set of orders; this is a hierarchical relationship implicit in the path:

```
/customer/{customer_id}/orders/{order_id}
```

When a customer is deleted, all their orders are also deleted, and this behavior is indicated by the **`mutual`** value of the deletion property.

A different relationship applies to books: a book cannot be deleted until all the orders which refer to it have been deleted, so deletion is indicated as **`disabled`** in that case.

There are three possible values for the **`dependee_deletion`** property:

- **`enabled`**: a referenced resource can be deleted even while dependent resources that refer to it still exist
- **`disabled`**: a referenced resource cannot be deleted until all dependent resources that refer to it have been deleted
- **`mutual`**: deleting a referenced resource deletes all the dependent resources that refer to it: this is typical where the dependent is owned by or contained in the referent

Pure resources

A pure resource is one that has no instances, or at least no instances that can be tracked with unique IDs, and is used as a placeholder for operations that do not behave as resource methods.

A pure resource differs from a standard resource in the following ways:

- it has no ID field (because there are no instances)
- dependencies have no **`dependee_deletion`** (for the same reason)
- all operations are grouped under a single **`pure`** category

For example:

```
ServiceAvailability:
  schemas:
    primary:
      json_ptr: '#/definitions/ServiceAvailability'
    operations:
```

```
    pure:
      - json_ptr: '#/paths/~1service~1status/get'
```

ServiceAvailability is a type that is returned as the result of a service status request but has no unique, persistent state.

Properties

The **properties** section assigns semantic categories to schema properties occurring in operation payloads, to constrain the range of values interpolated into generated request bodies.

For example:

```
properties:
  - json_ptr: '#/definitions/CardPayment'
    items:
      - name: card_number
        semantic: credit_card_number
      - name: expiry_date
        semantic: expiry
      - name: security_code
        semantic: cvv
```

The JSON Pointer expression locates a schema named **CardPayment** in the API document under **definitions**; the schema must have **object** type with at least the properties **card_number**, **expiry_date** and **security_code** which are specified here to have particular semantics that will guide the auto-generation of appropriate values wherever a **CardPayment** occurs in a request payload.

Semantic categories must be drawn from the following set:

address	cvv	latitude	state_code
age	date	longitude	street
area_code	domain	minutes	temperature
birthday	email	month	time
certificate	expiry	name	timestamp
charset	first_name	number	token
cidr	gender	paragraph	twitter
city	geo_location	percentage	url
color	hours	phone	user_agent
content_encoding	humidity	phone_number	username
content_type	iban	prefix	uuid
coordinates	id	pressure	version
country	identity_provider	price	year
country_code	ip_address	revision	zip_code
credit_card_number	language	sentence	
currency	language_code	social_security_number	
currency_code	last_name	state	

Deleting an AutoTest profile

To delete an AutoTest profile in a test suite:

1. Log in to the API Manager.
2. Click the **Test APIs** tile.
3. On the Test page, select a test suite.
4. Go to the **Tests** > **AutoTest Profiles** tab.
5. On the **Tests** > **AutoProfiles** tab, locate the AutoTest profile you want to delete, click the **Options** icon > **Delete** option.

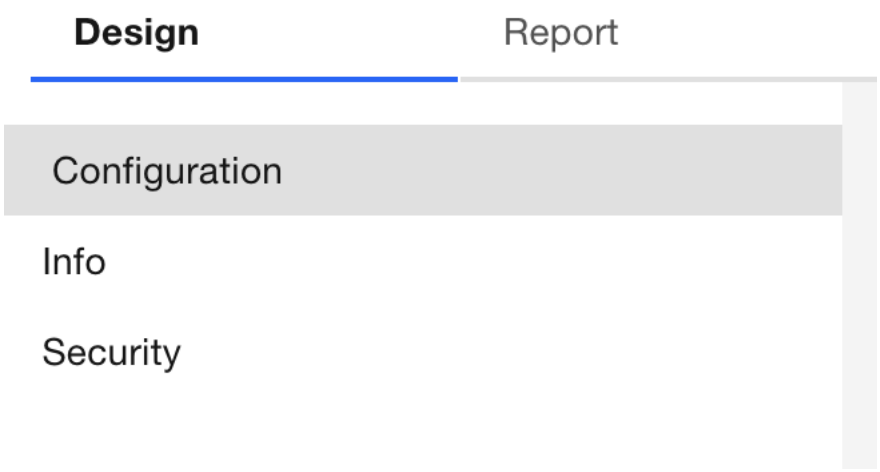
What to do next

- Next topic: Running an AutoTest profile

Running an AutoTest profile

To be able to run an AutoTest profile you will need to configure it first. Make sure to set the ServerURL parameter correctly otherwise all the generated API calls will fail. For more information please refer to AutoTest Profile Configuration. At any point during this process you can click on the save button to maintain your progress.

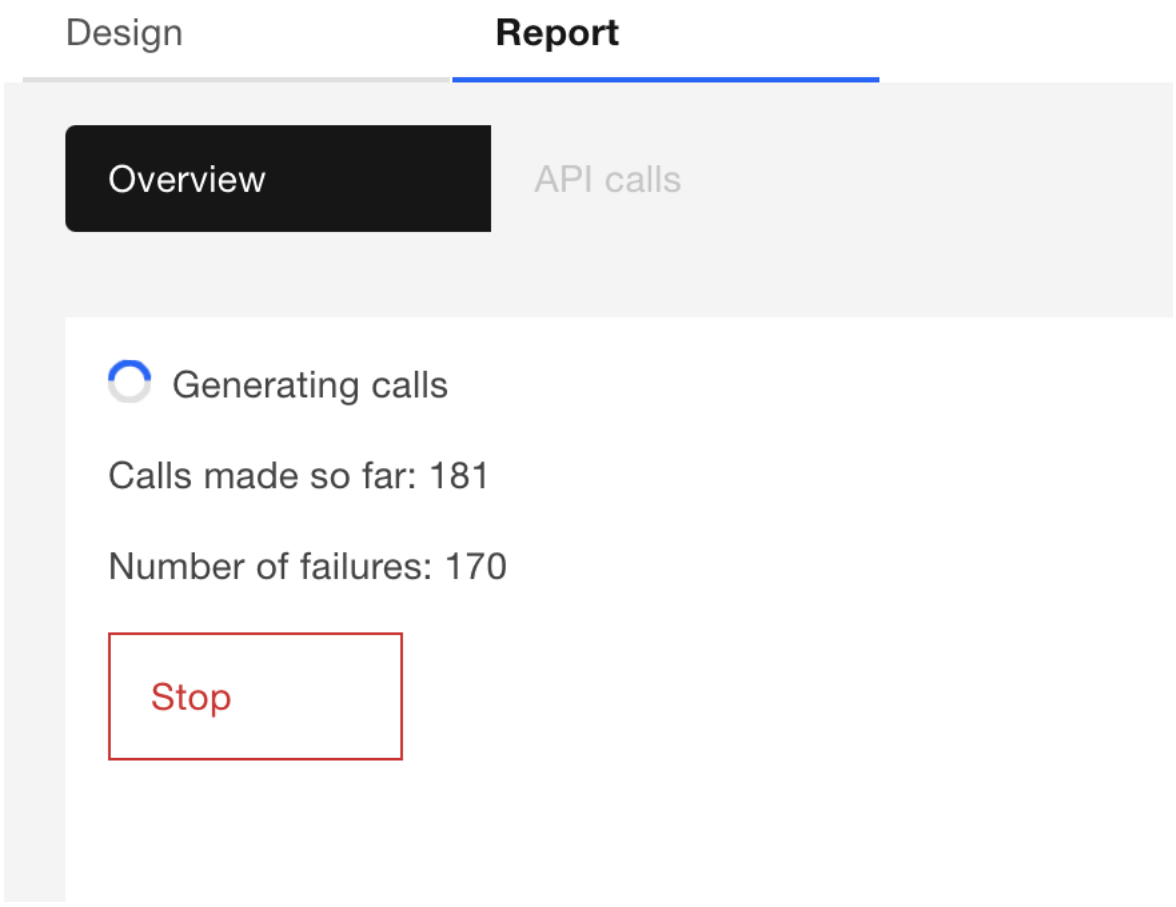
Once you have made your changes in the configuration panel, go to the security panel to set your authentication.



For more information read through AutoTest Profile Security.

Once you have made all your changes to the configuration and security. You can either save, or click on the run button.

This will take you to the report page where the exerciser will generate calls to test the API.



Once the generation has finished, it will take you to the next step AutoTest Report

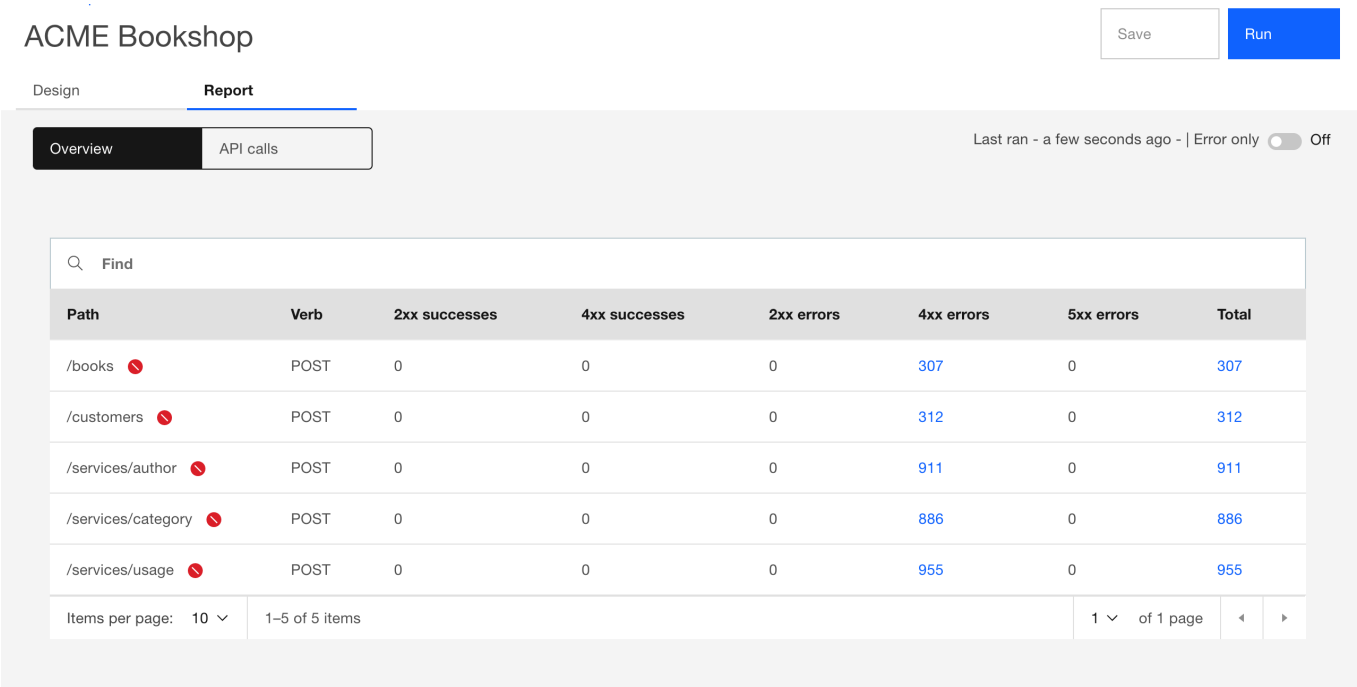
What to do next

- Next topic: Viewing the AutoTest Report

Viewing the AutoTest report

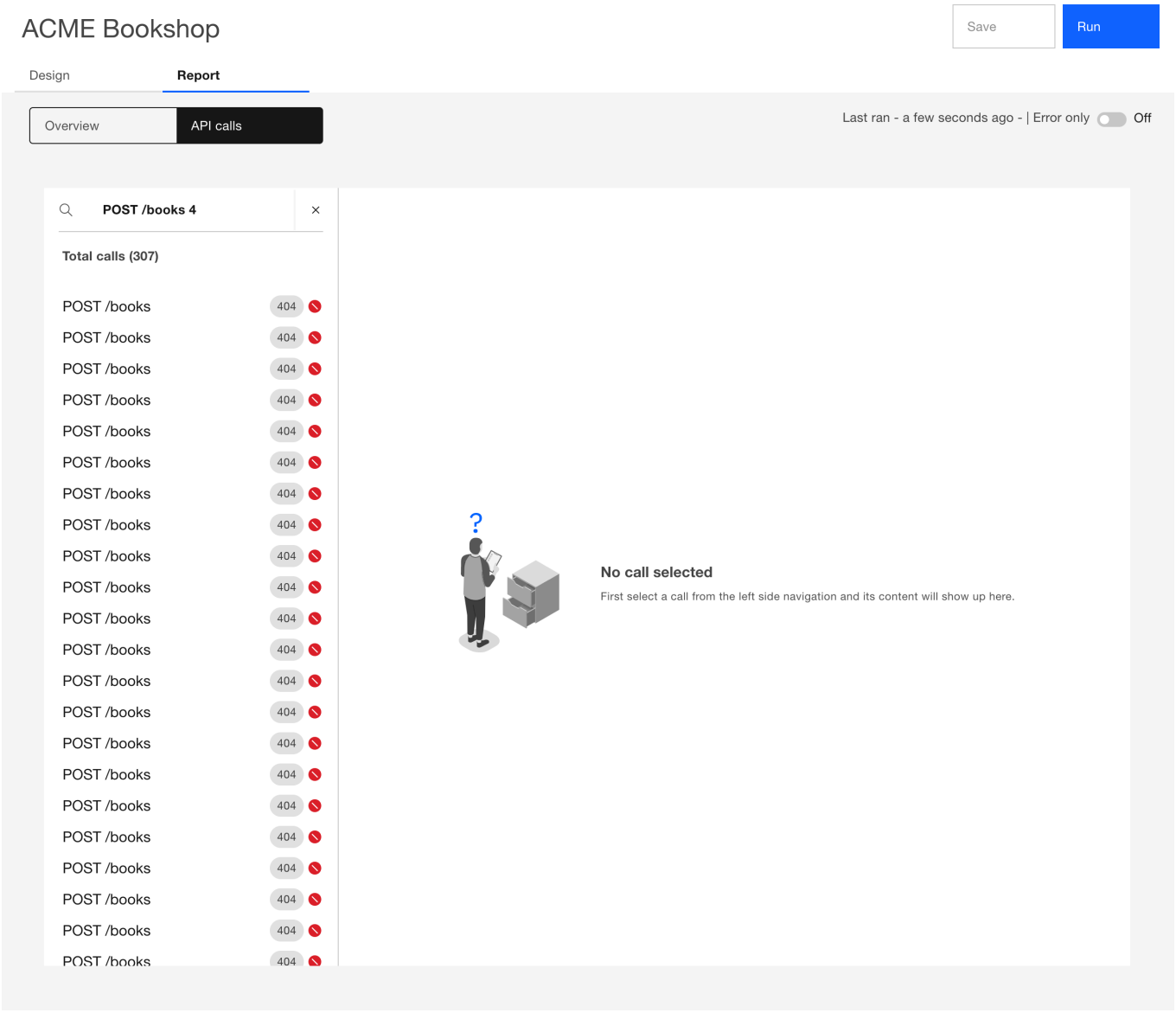
Once the AutoTest run has completed (either stopped manually or the configured time to run threshold has been reached) you will be presented with the **Report** section, where you will be able to view the results of the run.

First up is the **Overview** tab, which contains a table displaying all the paths defined in the AutoTest profile's API specification, alongside their associated HTTP methods (verbs), and followed by the number of each class of responses received throughout the course of the AutoTest profile run.



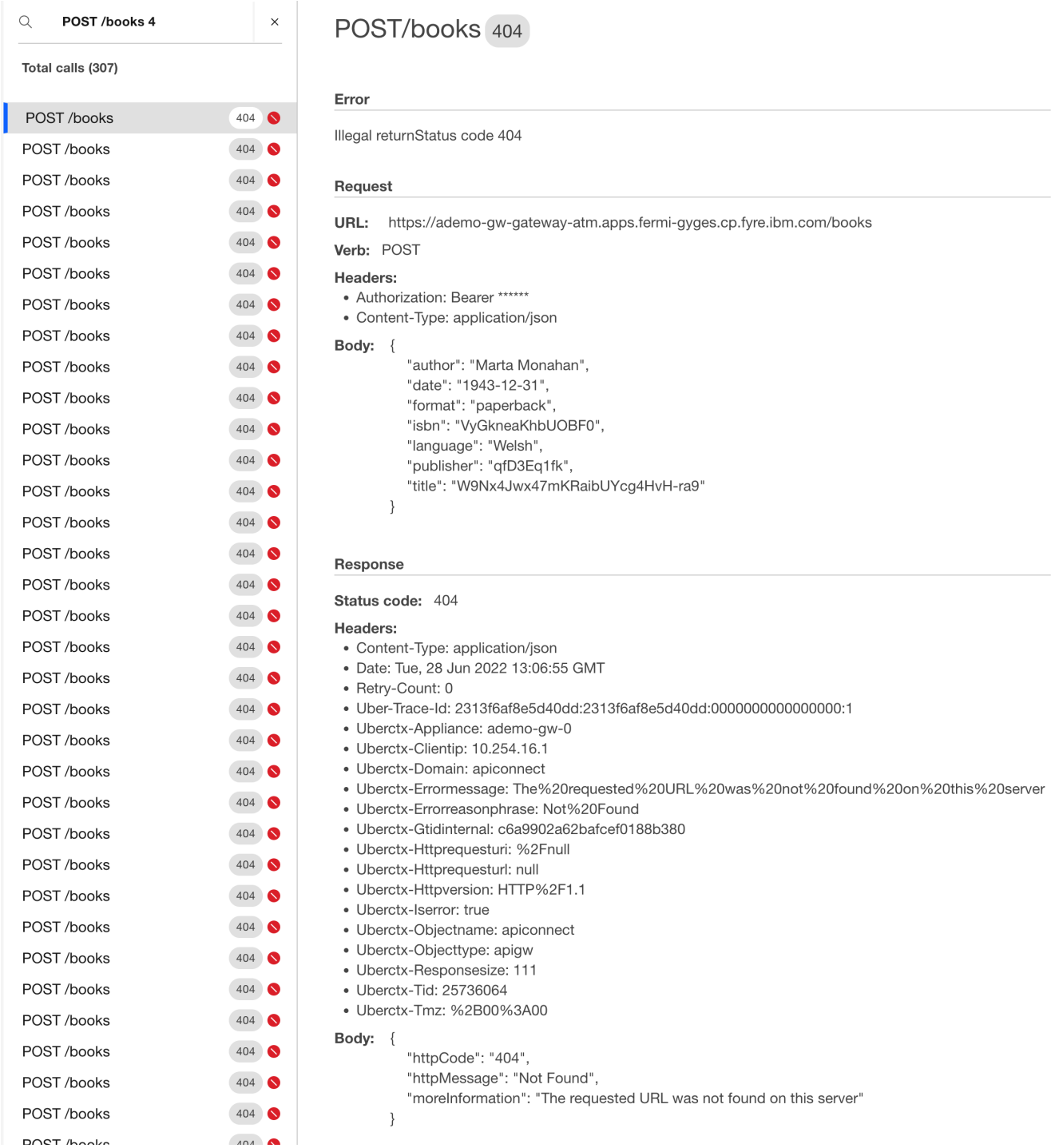
The **API calls** tab contains a more detailed view of every API call made by the run. The left side contains the list of the calls and their respective response statuses; the search bar lets you filter on the HTTP method, the path, and the response code; this should make it relatively easy to find specific requests amongst the potentially large number of API calls.

You can also get to the details view by clicking on any of the non-zero numbers in the overview table; this will automatically filter the list of API calls by the appropriate class of responses (or just the path and verb if the total tally is clicked on), as shown below.



*Either of the two views lets you apply a global **Errors only** filter by interacting with the toggle in the top right corner, under the **Save** and **Run** buttons. This will apply a filter to both the overview table and the detailed API calls list so that they will either display only errored or all requests.*

Selecting one of the calls from the list will render the details of that call on the right hand side. As seen below information such as headers and bodies for both the request and response is shown. This lets you closely exmaine precisely how a specific API endpoint had been called and the response the request received.

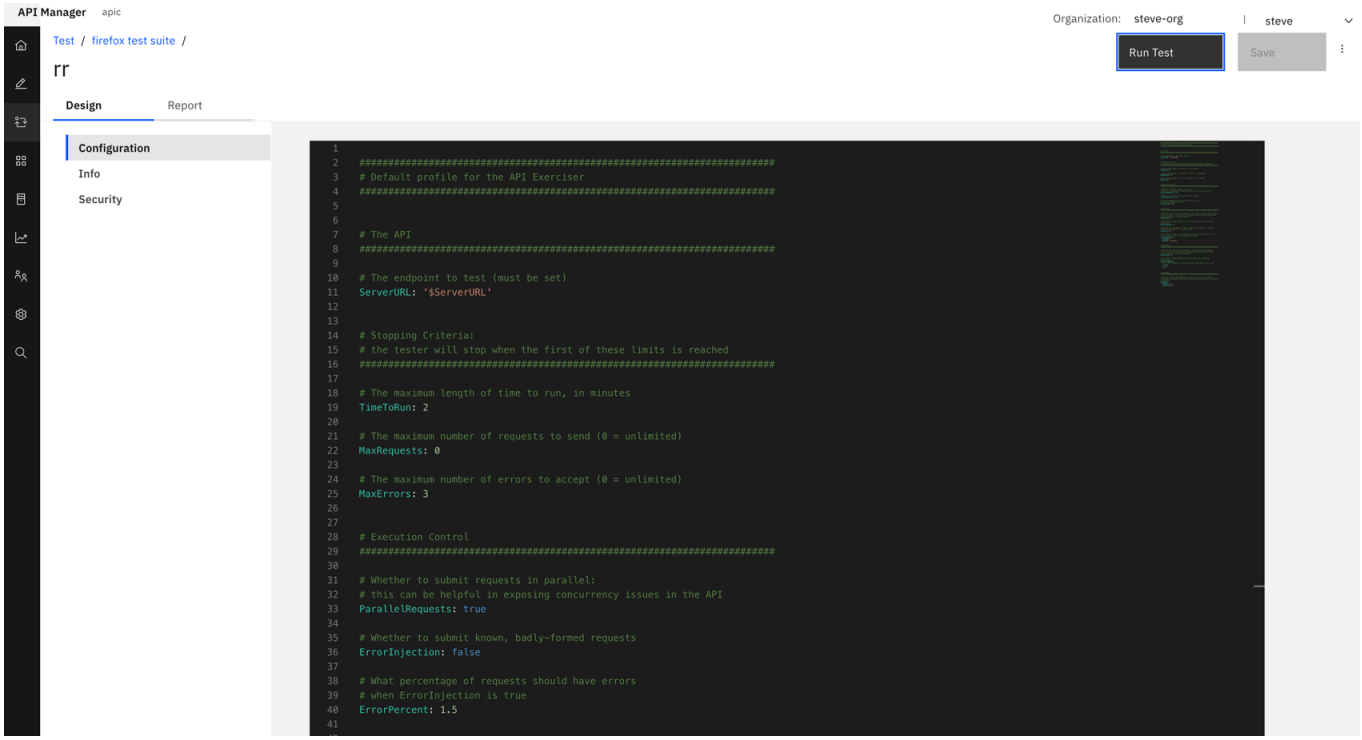


Profile Configuration

The configuration controls the execution of your profile; it determines for how long the test should run, and what kinds of requests it should generate. A newly-created profile includes a default configuration that you can run immediately to check that the auto-tester works with your API. The one thing you may have to set or change before you can run is the target Server URL which must point to your test deployment; the default value is copied from the API definition, where one is provided. You must also specify appropriate authentication settings on the separate **`Security`** tab (see Profile Security).

The configuration is in YAML format. The editor highlights syntactic errors as you type, and verifies the content for consistency with the API definition each time you press **`Save`**. If there is an error in the configuration, then the profile cannot run.

The Profile Configuration editor



Sections in the configuration:

- The API
- Stopping Criteria
- Execution Control
- Resources
- Operations
- Datatypes

The API

ServerURL

The server URL is the single endpoint to which all requests are sent. It must be a valid URL, reachable from your namespace, and include at least a scheme and a server address; it may also include a port number and a path prefix. It must not include a trailing slash. The server URL is prefixed to an operation path from the API definition to create the target of a request. For example, given this URL definition from the configuration:

```
ServerURL: https://api-host:8000/apis
```

And this operation path from the API definition:

```
paths:
  /movies:
    post:
```

The auto-tester will send **POST** requests to this target URL:

```
https://api-host:8000/apis/movies
```

Stopping Criteria

The profile stops running as soon as any of these criteria are met.

TimeToRun

The maximum length of time to run, in minutes: the auto-tester stops sending requests once this time period has elapsed. The overall running time will be longer because it includes additional time for setup and teardown.

For example, to set a maximum running time of ten minutes:

```
TimeToRun: 10
```

MaxRequests

The maximum number of requests to send: the auto-tester stops issuing requests once this number has been reached. The final total of requests may be greater if additional requests are needed for setup and teardown.

Set this to zero to disable the request limit, and then the auto-tester will issue as many requests as fit into the allotted time period.

For example, to set a maximum of two thousand requests:

```
MaxRequests: 2000
```

MaxErrors

The maximum number of error responses to allow: the auto-tester stops issuing requests once this number has been reached.

An error in this context is not an API error indicated by a **4xx** (or even, sometimes, a **5xx**) response, which is to be expected if the profile exercises all execution paths through the implementation. An error here is one where the response to a request is inconsistent with the API definition, which is a defect. Common cases include:

- the status code is not documented
- the payload does not conform to the schema
- any 500 error

There should be no such errors in a mature implementation so the limit is typically set low, so that the profile can terminate early if they occur. You can set the limit to zero and disable this stopping condition, which might be appropriate for a development environment where you know that some operations have defects (or you can disable those operations from ever being called as described under Operations).

For example, to set the error limit to three:

```
MaxErrors: 3
```

Execution Control

ParallelRequests

If this is **true** then the auto-tester spawns threads to issue multiple requests concurrently, otherwise it issues just one request at a time. The concurrent option increases throughput and can be helpful in exposing synchronization issues in the API.

The number of concurrent threads is fixed and cannot be changed.

For example, to disable concurrent requests:

```
ParallelRequests: false
```

ErrorInjection

If this is **true** then the auto-tester issues a certain percentage of malformed requests to further exercise the validation logic of the API, otherwise it only issues requests that are well-formed according to the API definition.

The percentage of requests is specified as **ErrorPercent**.

For example, to enable malformed requests:

```
ErrorInjection: true
```

ErrorPercent

The percentage of malformed requests, when **ErrorInjection** is **true**.

For example, to include 2% of malformed requests:

```
ErrorPercent: 2.0
```

Resources

The AutoTest execution model is based around the notion of "resources", where a resource has a schema (of object type) and a set of operations (or methods). Resources are defined in the API extensions.

The auto-tester only exercises operations that belong to resources. To decide which operation to execute next, the auto-tester first picks a resource type at random, and then an operation from that type (again, at random).

You control the frequency with which resources are selected by assigning them weights. The selection frequency for a given resource is computed as the weight for that resource divided by the sum of all the weights.

A zero weight means that the resource will never be selected, and you can use this to exclude certain resources from testing; for example, when you know that they are not yet fully implemented. It is an error for all weights to be zero which would leave nothing to test.

Use the **Resources** key to assign weights to specific resources, and **ResourceDefault** to set the default weight for any resources not specified.

Resources

The weights assigned to specific resource types. The value is an object, where the keys are resource names, and the values are non-negative integer weights. Resource names must be declared in the API extensions.

For example, to increase the test focus on **Customer** operations, and exclude **Order** operations entirely:

```
Resources:
  Customer: 2
  Order: 0
```

ResourceDefault

The default weight for any resource type not mentioned explicitly under **Resources**. The weight can be any non-negative integer value, but there are two common use cases:

- if the weight is 1 then all resources are selected with equal frequency, except where specified otherwise under **Resources**
- if the weight is 0 then all resources are disabled, except for those specified otherwise under **Resources**

For example, to disable all resources by default:

```
ResourceDefault: 0
```

In this case there must be at least one non-zero weight under **Resources**.

Instances

Upper and lower bounds on the number of resource instances that should exist at any time. The value is an object, where the keys are resource names, and the values specify lower and upper bounds for those resource types. Resource names must be declared in the API extensions, and the bounds must be non-negative integers that together define a non-empty range. The upper bound may also be **unlimited** in which case the number of instances depends upon the balance of create and delete requests issued for that type.

The auto-tester keeps track of the instances it creates for each resource type, in order to provide valid references and avoid too many 404 responses, and makes best efforts to keep the number of instances within any specified bounds.

For example, to place an upper bound on the number of **Customer** instances that should exist at any time:

```
Instances:
  Customer:
    maximum: 25
```

InstanceDefault

The default instance bounds for any resource type not mentioned explicitly under **Instances**.

For example, to make the default unlimited:

```
InstanceDefault:
  minimum: 1
  maximum: unlimited
```

Operations

Once the auto-tester has selected a resource type, it uses a similar weighting strategy to choose an operation from that type.

You control the frequency with which operations are selected by assigning them weights. Use the **Operations** key to assign weights to specific operations, and **OperationDefault** to set the default weight for any operations not specified.

A zero weight disables an operation entirely, and if all the operations are disabled for a resource then the resource itself is also disabled.

Operations

The weights assigned to specific operations. The value is an object whose schema follows the pattern established by the **Paths** object in OpenAPI, but the values assigned to each operation are non-negative integer weights.

For example, to increase test focus on the **put_customer** operation, and avoid **patch_customer** completely:

```
Operations:
  /customers/{customer_id}:
    put: 10
    patch: 0
```

The special key _ can be used to match all methods defined on a path that are not called out explicitly. For example, to disable all but the **put_customer** operation:

```
Operations:
  /customers/{customer_id}:
    put: 10
    _: 0
```

Remember that the weights are interpreted only across operations within the same resource type, and not across all operations in the API, so the overall distribution is determined by both the weight on the resource and the weight on the operation.

OperationDefault

The default weight for any operations not mentioned explicitly under **Operations**. The value is an object where the keys are method names and the values are the default weights for those methods where they occur under any path. The special key _ matches all method names that are not listed separately.

For example, to disable all operations by default:

```
OperationDefault:
  _: 0
```

In this case there must be at least one non-zero weight under **Operations**.

Datatypes

Place here any custom rules that guide the auto-tester in generating values for operation parameters and payloads. These rules override the default rules derived from the API definition and any semantic annotations in the API extensions.

Rule syntax is described in Data Generation Rules.

Schemas

Data generation rules for named schemas found under **definitions** (OpenAPI 2.0) or **components/schemas** (OpenAPI 3.0).

For example, given the schema definition:

```
Movie:
  type: object
  properties:
```

```
    title:
      type: string
    release_date:
      type: string
      format: date
    language:
      description: Two-letter ISO 639-1 language code
      type: string
      pattern: '[a-z]{2}'
```

The default generation rule for the **language** property, derived from this schema, will produce random string values that match the specified pattern (two lower-case alphabetic characters) but many of these will not be valid language codes, potentially giving rise to an excessive number of 400 responses.

To override the default generation rule and specify instead some particular language codes sufficient for testing:

```
Datatypes:
  schemas:
    Movie:
      properties:
        language:
          enum:
            - en
            - fr
            - de
```

To inject a small percentage of invalid values:

```
Datatypes:
  schemas:
    Movie:
      properties:
        language:
          choice:
            - enum:
                - en
                - fr
                - de
              weight: 98
            - const: xx
              weight: 2
```

Operations

Data generation rules for operation parameters and payloads described by inline (or unnamed) schemas and types. The value is an object whose schema follows the pattern established by the **Paths** object in OpenAPI.

Note that the auto-tester only supports **path** and **query** parameters at present, and other parameter types are ignored.

For example, given these (abbreviated) operation definitions:

```
paths:
  /movies:
    get:
      parameters:
        - name: limit
          in: query
          description: Sets an upper bound on the number of movies returned
          schema:
            type: integer
    post:
      requestBody:
        content:
          application/json:
            schema:
              type: object
              allOf:
                - $ref: '#/definitions/Movie'
                - properties:
                    keywords:
                      type: array
                      items:
                        type: string
```

To set values for the **limit** query parameter, and the **keywords** property of the payload to **post**:

```
Datatypes:
  operations:
    /movies:
      get:
        parameters:
          - name: limit
            in: query
            data:
              # pick a random value between 0 and 500 (inclusive)
              minimum: 0
              maximum: 500
      post:
        requestBody:
          content:
            application/json:
              data:
                properties:
                  keywords:
                    # pick up to 3 items from the following list
                    items:
```

```
                    enum:
                      - avant-garde
                      - dystopia
                      - epic
                      - postmodern
                      - remake
                      - shootout
                  minItems: 0
                  maxItems: 3
```

Profile Security

Security is where you set credentials for the API under test. The auto-tester supports only **Basic** authentication (username and password) and **Bearer** (or **Token**) authentication; it does not support other authentication methods such as **OAuth,** or the use of custom headers.

Select your security type from the dropdown and enter the required information:

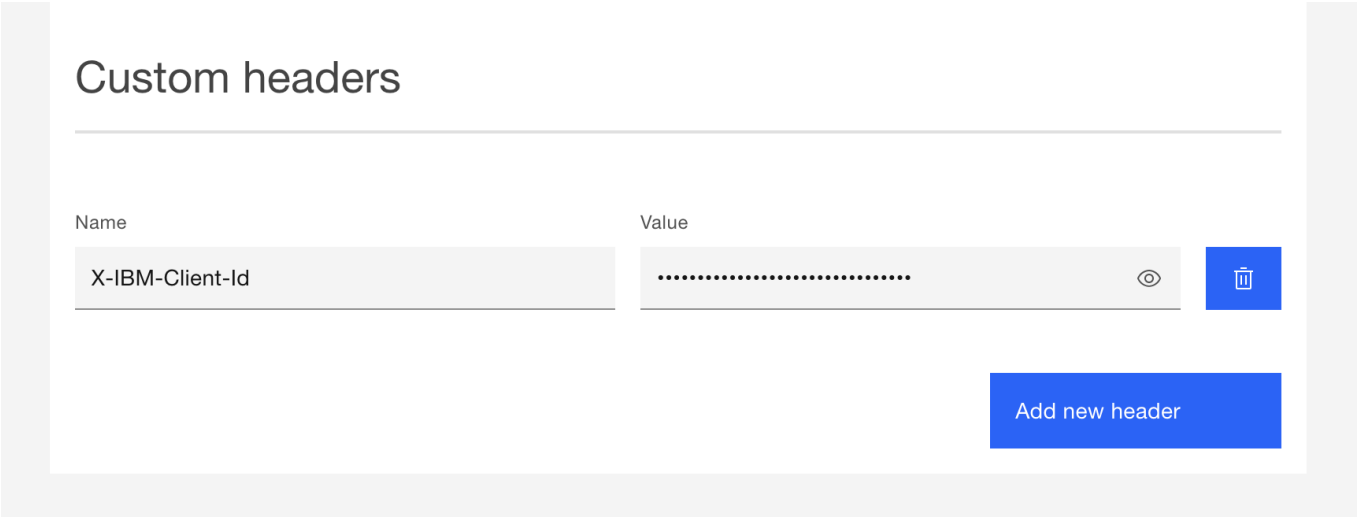
Security Type	Description	Credentials
None	Unsecured endpoint	n/a
Basic	Basic HTTP authentication	Username/Password
Basic-url	Same as **Basic** but omits the word `Basic` from the Authorization header	Username/Password
Bearer	Bearer (or Token) authentication	Token

If you provide incorrect credentials then they will not prevent the profile from running, but you can expect to see all requests fail, typically with the response **401 Unauthorized** or **403 Forbidden.**

If you provide a bearer token then be sure that it has a sufficient lifetime to cover the full period of the test, which is determined by the stopping criteria in the profile configuration. Otherwise, once the token expires, you can expect to see authentication failures as described earlier.

Custom headers

You can also confiure custom security headers should your APIs require them for authentication/authorization. Simply add the required header(s) and the appropritate value(s) under the custom headers section like so:



Data Generation Rules

This feature is available only in API Connect Enterprise as a Service

Data generation rules guide the auto-tester in choosing values to populate the parameters and payloads of its requests.

There are default rules derived from the API definition and extensions, but sometimes you want to override these, either to improve the general accuracy of the auto-tester or to tailor its requests to a specific use case, and you do that by adding custom rules to the Datatypes section of the profile configuration.

A data generation rule binds a **data generator** to a schema or type in the API definition such that, when the auto-tester assembles the data for a request and encounters that schema or type, it uses the generator to provide an appropriate value at that point.

Data generators

A **data generator** describes a never-ending sequence of values, and whenever the auto-tester calls on a generator it takes the next value from the sequence.

The syntax for generation rules reuses terms from JSON Schema where the implied constraints on a schema are sufficient to guide the auto-tester in generating conformant values. Such constraints occurring in the API definition help inform the default rules.

Constant

The **constant** generator always returns the same value, which can be of any simple type.

```
const: 0

# ==> 0, 0, 0, ...

const: Thursday

# ==> "Thursday", "Thursday", "Thursday", ...
```

Enumeration

The **enumeration** generator returns values selected at random from a given list, which can be of any simple type.

```
enum:
  - 0
  - 1
  - 2

# ==> 1, 0, 2, 0, ...

enum:
  - Thursday
  - Friday
  - Saturday

# ==> "Friday", "Saturday", "Saturday", ...
```

Pattern

The **pattern** generator returns string values that match a given regular expression.

```
pattern: '[0-9]{3}'

# ==> "884", "924", "121", ...

pattern: '[a-z][a-z_]{0,15}'

# ==> "dbe", "kg_cfsv", "dhbfgbjxavmsaxq", ...
```

Range

The **range** generator returns numeric values uniformly distributed over a given range (inclusive).

```
minimum: 0
maximum: 10

# ==> 10, 6, 8, ...
```

If either endpoint is omitted then it defaults to the most negative or most positive value of the type:

```
minimum: 0

# ==> 10029625, 3162517758, 841054690, ...
```

Semantic

The **semantic** generator returns values that conform to some pre-defined semantic category, covering names, addresses, credit card numbers, etc. The set of categories is the same as that allowed in the API extensions.

```
semantic: email

# ==> "herbertschaden@fahey.net", "emmyfarrell@cruickshank.com", "vilmaprohaska@zboncak.org", ...

semantic: first_name

# ==> "Antwan", "Connie", "Modesto", ...
```

Resource

The **resource** generator returns values that are unique identifiers of resource instances of a given type, selected at random from the pool of existing instances. The identifier of an instance is taken from the `id_name` property of the resource type defined in the API extensions and its type and format will vary accordingly.

```
resource: Customer

# ==> "e5537298-9041-47fc-b97e-c910d8f9d971", "f60521c8-2ccb-4036-a742-9f350271675f", "e5537298-9041-47fc-b97e-c910d8f9d971",
...
```

Array

The **array** generator returns array values whose contents are populated from a data generator defined recursively. The length of the array is chosen at random between `minItems` and `maxItems` inclusive.

```
items:
  pattern: '[0-9]{3}'
minItems: 3
maxItems: 3

# ==> ["770", "496", "219"], ["431", "895", "331"], ["864", "130", "204"], ...

items:
  enum:
    - 0
    - 1
minItems: 0
```

```
maxItems: 5

# ==> [], [1, 0, 1, 1], [0], ...
```

Object

The **object** generator returns object values whose properties are populated from named data generators, defined recursively.

```
properties:
  x:
    minimum: 0
    maximum: 100
  y:
    minimum: 0
    maximum: 100

# ==> {"x": 59, "y": 63}, {"x": 22, "y": 8}, {"x": 95, "y": 57}, ...
```

```
properties:
  code:
    pattern: '[0-9]{3}'
  language:
    enum:
      - en
      - fr
      - de

# ==> {"code": "452", "language": "fr"}, {"code": "504", "language": "en"}, {"code": "846", "language": "fr"}, ...
```

If a property is not required by the target schema then it may be marked with an **optional** attribute which specifies a frequency (between **0.0** and **1.0**) that determines how often the property is included in the generated object (where 0 means never, and 1 means always).

```
properties:
  code:
    pattern: '[0-9]{3}'
  language:
    optional: 0.5
    enum:
      - en
      - fr
      - de
  tag:
    optional: 0.0

# ==> {"code": "452", "language": "fr"}, {"code": "504", "language": "en"}, {"code": "846"}, ...
```

Choice

The **choice** generator returns values selected at random from a given list of data generators, using optional weights to bias the outcome. The alternatives must be simple generators, which excludes **array** and **object** generators.

```
choice:
  # bias the selection towards English
  - const: en
    weight: 5
  - const: fr
  - const: de

# ==> "en", "en", "en", ..., "de", "en", ...
```

```
choice:
  # normal values in the range 1-10
  - minimum: 1
    maximum: 10
    weight: 98
  # with a small percentage of outliers
  - const: 999
    weight: 2

# ==> 6, 3, 9, 1, ..., 999, 6, ...
```

Data generation rules

A **data generation rule** binds a data generator to a schema occurring in the API definition so that the auto-tester knows how to generate values for that schema in an API call. Rules are specified in the Datatypes section of the profile configuration, and override the default rules derived from the API definition and extensions.

Schemas occur widely in the OpenAPI specification, but there are three key locations which have most relevance to the auto-tester where the profile allows override rules:

- Named schemas
- Parameters
- Inline request bodies

Named schemas

A schema rule assigns a data generator to a named schema from the API definition, found either under **definitions** (OpenAPI 2.0) or **components/schema** (OpenAPI 3.0). The auto-tester will use the specified generator wherever the named schema occurs in a request.

Schema rules appear in the schemas section of the profile configuration, under **Datatypes**.

Because this is an *override,* the generator may be incomplete for the original schema, and the auto-tester will fall back to using the default rules to cover any gaps. In particular, if the schema describes an object type then the generator need only name a subset of the object properties, and the auto-tester will apply the override for those properties but use default rules for the rest.

For example, given this named schema in the API definition:

```
Movie:
  type: object
  properties:
    title:
      type: string
    release_date:
      type: string
      format: date
    language:
      description: Two-letter ISO 639-1 language code
      type: string
      pattern: '[a-z]{2}'
```

The default rule derived from the schema returns values such as:

```
{
  "title": "mkLHM22Ohabb6YG4-wdZFUwxQa7dn",
  "release_date": "2018-09-14",
  "language": "jq"
}
```

You can refine this by overriding the default rules for the **title** and **language** properties:

```
Datatypes:
  schemas:
    Movie:
      properties:
        title:
          semantic: sentence
        language:
          enum:
            - en
            - fr
            - de
```

This rule binds an object generator (introduced by the keyword **properties**) to the schema named **Movie** which -- combined with the default rule for **release_date** -- returns values such as:

```
{
  "title": "What dog everybody am myself hourly meeting how group over example her",
  "release_date": "2094-06-31",
  "language": "de"
}
```

Parameters

A parameter rule binds a data generator to a named parameter within an operation found under **paths** in the API definition.

Parameter rules appear in the operations section of the profile configuration, under **Datatypes**.

For example, given this parameter definition on **GET /movies** in the API definition that restricts the number of movies returned in a single call:

```
paths:
  /movies:
    get:
      parameters:
        - name: limit
          in: query
          description: Sets an upper bound on the number of movies returned
          schema:
            type: integer
```

The default rule derived from this definition will return values from the full integer range, but you can override this with a rule in the profile configuration that specifies a more realistic range, with occasional outliers to check the validation logic:

```
Datatypes:
  operations:
    /movies:
      get:
        parameters:
          - name: limit
            data:
              choice:
                - minimum: 1
                  maximum: 500
                  weight: 95
                - const: 0
                  weight: 3
                - const: 8000000
                  weight: 2
```

The format of the rule follows the format of the original definition but with the keyword **data** in place of **schema** to introduce the data generator.

In OpenAPI, a parameter is uniquely identified by a name *and a location* (so the same name could be used in, say, both query and header) so the parameter rules allow the **in** property as optional where needed to disambiguate. It's not used in this example because the query parameter is the only one named **limit**.

The auto-tester only supports parameters of type **query** and **path** at present (and **body** in OpenAPI 2.0). Rules bound to other parameter types are ignored.

Request body

A request body rule binds a data generator to the body of a request where the corresponding schema is specified directly as part of the operation in the API definition, rather than via a reference to a named schema.

Request body rules appear in the operations section of the profile configuration, under **Datatypes**.

For example, given this definition of the payload to a **POST** request:

```
paths:
  /movies:
    post:
      requestBody:
        content:
          application/json:
            schema:
              type: object
              allOf:
                - $ref: '#/components/schemas/Movie'
                - properties:
                    keywords:
                      type: array
                      items:
                        type: string
```

The following rule generates some appropriate keywords to attach to a new movie (where the movie itself is covered by the rules discussed earlier):

```
Datatypes:
  operations:
    /movies:
      post:
        requestBody:
          content:
            application/json:
              data:
                properties:
                  keywords:
                    # pick up to 3 items from the following list
                    items:
                      enum:
                        - avant-garde
                        - dystopia
                        - epic
                        - postmodern
                        - remake
                        - shootout
                    minItems: 0
                    maxItems: 3
```

As with the parameter rules, the format follows that of the original definition but with **data** in place of **schema**. The definition is in the style of OpenAPI 3.0, but the rule can work for OpenAPI 2.0 as well as long as the content type is consistent; alternatively, starting from an OpenAPI 2.0 definition, you can use a **body** parameter rule to specify the payload, but that does not work for 3.0.

Testing an API with the Policies editor

IBM® API Connect provides a basic test environment in the Policies editor so that you can ensure that your APIs are defined and implemented correctly.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note:

- If you are testing an OpenAPI 2.0 API, you can use either the Policies editor test tool described here, or you can use the Test tab. If you are testing an OpenAPI 3.0 API, only the Test tab is available; see Using the Test tab to debug your API.
- If you are testing an API that contains references to API properties, only those references that are defined inside the API assembly are resolved and replaced with their corresponding values when you invoke the API in the assembly test tool; property references that are defined outside of the API assembly are not resolved. For more information on API properties, see Setting API properties (OpenAPI 2.0) or Setting API properties (OpenAPI 3.0).
- Due to Cross-Origin Resource Sharing (CORS) restrictions, the assembly test tool cannot be used with the Chrome or Safari browsers on the macOS Catalina platform.

Procedure

To test an API, complete the following steps.

1. If you are using API Designer, set the mode to Online using the Options menu ⋮ on the main page.
2. In the navigation pane, click Develop, then select the APIs tab.
3. Click the title of the API that you want to work with.
4. Click the Gateway tab then, in the navigation pane, click Policies.
5. If you are testing the API for the first time and, when you created the API definition, you selected the Activate API option, your test setup will already be configured and you can proceed immediately to the next step to test your API. Otherwise, click Offline to toggle the API to the Online state. If you are retesting your API after making changes, the modified API will have been republished automatically on saving.
6. Click the Test icon ▷.
   Note:
   - The Test icon is active only if you have toggled your API to the Online state, as described in step 5.
   - The Catalog, Product, Plan, and application details displayed in the Setup pane reflect your configured API testing preferences; see Specifying the testing preferences for an API,

7. In the Operation section, select the API operation that you want to test, then click Invoke.
   The API response is displayed in the Response section.
   Note: If you receive a message relating to an untrusted certificate, click the link provided, accept the certificate, then return to the test environment and click Invoke again. The message also mentions a lack of CORS support on the server, but this is just one possible cause for the connection failing.

Testing an API with the Local Test Environment

Use the Local Test Environment to test APIs on your local machine, without the need to connect to an API Connect management server. The Local Test Environment is a lightweight API Manager running on your local machine. It allows you to rapidly test APIs locally.

API Connect provides the following methods for testing an API on your local machine:

- Invoke the API from the API Designer UI application running in Online mode as described in Testing an API.
- Call the API in the Local Test Environment with a cURL command, as described in the following sections.

Prerequisites

- The API Connect developer toolkit, including the API Designer user interface, installed. For installation and running instructions, see Installing the toolkit.
- The Local Test Environment and the API Designer must be from the same API Connect fix pack release in order for them to work together.
- Docker installed.
  Note: The Local Test Environment is not supported with Docker Version 18.09.x.
- A minimum of 4 GB of RAM available to Docker if a single gateway type is used, or 6 GB if both the DataPower® API Gateway and DataPower Gateway (v5 compatible) are used.
  Note: As you increase the number of APIs that are published to your gateways, you will need to allocate further memory to Docker. You will also need to start the Local Test Environment with a larger database; see **`apic-lte start`**.
- If you are using Windows, ensure that your C: drive (or the drive on which your HOME directory is located, if different), is enabled as a shared drive so that the Local Test Environment files are accessible to the Docker containers.

Installing the Local Test Environment

There are two options for installing the Local Test Environment:

- Each user downloads the Local Test Environment images to their local machine and installs the Local Test Environment from there.
- One user downloads the Local Test Environment images and uploads them to a private Docker registry, from where any user can install the Local Test Environment.

To install the Local Test Environment from your local machine, complete the following steps:

1. Open a browser and visit the API Connect announcement page, search for your product version, and locate the "Downloads" table on the product announcement page. In the table, click the link to download each of the following files:
   - apic-lte-images-*version*.tar.gz, which contains all required Docker images.
     Example: apic-lte-images-10.0.2.tar.gz
   - apic-lte-*platform*-*version*, which are binary files for the Mac OS X, Linux®, and Windows platforms.
     Examples:
     - Mac OS X: apic-lte-osx-10.0.2
     - Linux: apic-lte-linux-10.0.2
     - Windows: apic-lte-win-10.0.2
2. On the Mac OS X or Linux platform, use the **`chmod`** command to make the binary file executable; for example:

   ```
   chmod +x linux-apic-lte
   ```

3. Load the Docker images into your local Docker image repository by entering the following command:

   ```
   docker load < apic-lte-images.tar.gz
   ```

Note: In all Local Test Environment commands that are used on the remainder of this page, replace *platform* with **`osx`**, **`linux`**, or **`windows`**, depending on your platform, as follows:

- Mac OS X: replace ***`platform`*`-apic-lte`** with **`osx-apic-lte`**
- Linux: replace ***`platform`*`-apic-lte`** with **`linux-apic-lte`**
- Windows: replace ***`platform`*`-apic-lte`** with **`win-apic-lte`**

To upload the Local Test Environment images to a private Docker registry, complete the following steps:

1. Download the **`IBM_API_CONNECT_LOCAL_TEST_ENVIRO.zip`** file as described in step 1.
2. Distribute the appropriate binary file to all users, according to the platform.
3. Upload the Local Test Environment to your private Docker registry; enter the following command:

   ```
   platform-apic-lte registry-upload apic-lte-images.tar.gz registry_host
   ```

   where *registry_host* is the host name or IP address of your private Docker registry. Now, any user can install and run the Local Test Environment as follows:
   a. If the private Docker registry requires authentication, log in by entering the following command:

      ```
      docker login registry_host
      ```

   b. Load the Docker images into your local Docker image repository by entering the following command:

      ```
      platform-apic-lte init registry_host
      ```

Starting the Local Test Environment

1. Start the Docker images by running the following command:

   ```
   platform-apic-lte start
   ```

   Note:
   - By default, the `platform-apic-lte start` command starts only a DataPower API Gateway. To also start a DataPower Gateway (v5 compatible), enter the following command:

     ```
     platform-apic-lte start --datapower-gateway-enabled --datapower-api-gateway-enabled
     ```

   - The Local Test Environment might fail to start with an error message that includes the strings `Error: certificate is not yet valid` and `CERT_NOT_YET_VALID`. The most likely cause is that the date and time setting is incorrect on the machine that is the running the Local Test Environment. Ensure that the date and time setting is correct, before attempting the start command again. If you are using Docker for Windows, the clock in the Docker containers can become out of sync with the system clock, especially after a machine has been put in sleep mode. In this case, restarting Docker should fix the clock discrepancy; for more information, see https://github.com/docker/for-win/issues/4526.
   - By default, the LTE starts with an empty backend database that does not contain the APIs and Products that might have been published during an earlier run of the LTE. To start the LTE with the backend database that was used during the previous run, use the flag `--keep-config`, for example, `platform-apic-lte start --keep-config`. When you use `--keep-config` any other flag that is specified for the start is ignored. Instead the same flags that were used during the earlier start are used, in particular the same gateways are enabled.
2. Verify that the Local Test Environment is installed and running correctly by running the following commands:
   a. Check the status of the LTE components:

      ```
      platform-apic-lte status
      ```

      This output from this command shows the status of all components, and provides endpoint and authentication details, and should be similar to the following:

      ```
      Container                       Status
      ---------                       ------
      apic-lte-apim                   Up 3 minutes
      apic-lte-datapower-gateway      Not Running
      apic-lte-datapower-api-gateway  Up 2 minutes
      apic-lte-db                     Up 3 minutes
      apic-lte-juhu                   Up 3 minutes
      apic-lte-lur                    Up 3 minutes

      - Platform API url: https://localhost:2000
      - Admin user: username=admin, password=7iron-hide
      - 'localtest' org owner: username=shavon, password=7iron-hide
      - 'localtest' org sandbox test app credentials client id: 80963e74076afe50d346d76401c3c08a
      - Datapower API Gateway API base url: https://localhost:9444/localtest/sandbox/
      ```

   b. Log in to the management server:

      ```
      apic login --server localhost:2000 --username shavon --password 7iron-hide --realm provider/default-idp-2
      ```

      This command confirms that you can log in to the management server, and the response should be as follows:

      ```
      Logged into localhost:2000 successfully
      ```

      Known issue: If you receive an error during the login, stop and then restart the LTE; then log in again.

Note: If you have previously installed toolkit credentials, as detailed in Installing the toolkit, attempting to log in to the Local Test Environment management server from the toolkit CLI will fail; you must first clear the credentials by using the following command:

```
apic client-creds:clear
```

The toolkit CLI will now use the default credentials for login to the toolkit Local Test Environment. If you subsequently log in to a management server in an API Connect cloud, the default toolkit credentials will also be used, rather than the credentials that are unique to that deployment. To use the unique credentials, re-install them by using the following command:

```
apic client-creds:set toolkit_credentials_file_path/credentials.json
```

where *credentials_file_path* is the location of the previously downloaded toolkit credentials JSON file.
To determine whether you currently have any toolkit credentials installed, use the following command:

```
apic client-creds:list
```

Preparing an API for testing in the Local Test Environment

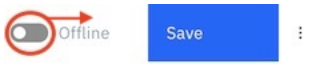
To prepare an API for testing in the Local Test Environment, you must publish it to the Sandbox Catalog in the Local Test Environment. If you want to test an API that you already published, proceed to Testing an API in the Local Test Environment., otherwise, complete the following steps:

1. Launch the API Designer user interface.
2. Open the required local directory; this is the directory in which your API and Product definition files will be stored.
3. Connect to the Local Test Environment. If you haven't previously connected to the Local Test Environment, click Add Another Cloud, then complete the following steps:
   a. In the HOST URL field, enter `https://localhost:2000`, then click Next.
   b. In the Username field, enter `shavon`, in the Password field enter `7iron-hide`, then click Sign in.
      Note: If you are not able to connect to the Local Test Environment using `https://localhost:2000`, enter `https://127.0.01:2000` instead in the HOST URL field.
      Known issue: If you receive an error during the login, stop and then restart the LTE; then log in with API Designer again.

   If you have previously connected to the Local Test Environment, click the existing tile to log in immediately.

   The API Designer welcome page opens.

4. Click Develop APIs and Products, then click the API that you want to test. For details on how to configure an API definition, see Developing your APIs and applications.
5. Move the activation slider control to the on position:



When the publish operation completes, your API is ready for testing.
Note: Whenever you make any changes to an API, you must republish it before retesting.

Testing an API in the Local Test Environment.

To test an API in the Local Test Environment, issue a REST API call to the following URL:

```
https://localhost:9444/localtest/sandbox/basepath/operation_path?client_id=lte_client_id
```

where:

- *basepath* is the base path that is configured in the API definition.
- *operation_path* is the path for the operation that you want to invoke, as configured in the API definition.
- *lte_client_id* is the client ID for the test application in the local test environment, as returned by the **platform-apic-lte status** command in step 2.

The following example show how to test the API that is created in the tutorial Creating a proxy REST API definition, by using the **curl** utility; the API returns the details of bank branches:

```
curl -k https://localhost:9444/localtest/sandbox/branches/details?client_id=80963e74076afe50d346d76401c3c08a
[{"id":"0b3a8cf0-7e78-11e5-8059-a1020f32cce5","type":"atm","address":{"street1":"600 Anton Blvd.","street2":"Floor
5","city":"Costa Mesa","state":"CA","zip_code":"92626"}},
{"id":"9d72ece0-7e7b-11e5-9038-55f9f9c08c06","type":"atm","address":{"street1":"4660 La Jolla Village Drive","street2":"Suite
300","city":"San Diego","state":"CA","zip_code":"92122"}},
{"id":"ae648760-7e77-11e5-8059-a1020f32cce5","type":"atm","address":{"street1":"New Orchard
Road","city":"Armonk","state":"NY","zip_code":"10504"}},
{"id":"c23397f0-7e76-11e5-8059-a1020f32cce5","type":"branch","phone":"512-286-5000","address":{"street1":"11400 Burnet
Rd.","city":"Austin","state":"TX","zip_code":"78758-3415"}},
{"id":"ca841550-7e77-11e5-8059-a1020f32cce5","type":"atm","address":{"street1":"334 Route
9W","city":"Palisades","state":"NY","zip_code":"10964"}},
{"id":"dc132eb0-7e7b-11e5-9038-55f9f9c08c06","type":"branch","phone":"978-899-3444","address":{"street1":"550 King
St.","city":"Littleton","state":"MA","zip_code":"01460-1250"}},
{"id":"e1161670-7e76-11e5-8059-a1020f32cce5","type":"branch","phone":"561-893-7700","address":{"street1":"5901 Broken Sound
Pkwy. NW","city":"Boca Raton","state":"FL","zip_code":"33487-2773"}},
{"id":"f9ca9ab0-7e7b-11e5-9038-55f9f9c08c06","type":"atm","address":{"street1":"1 Rogers
Street","city":"Cambridge","state":"MA","zip_code":"02142"}}]
```

Local Test Environment commands

The following table summarizes the Local Test Environment commands; use the **help** command to get full usage details for any command.

Table 1. Local Test Environment command summary

Command	Description
platform-apic-lte help *command*	Display help information for any command.
platform-apic-lte init	Download the Local Test Environment Docker images.
platform-apic-lte start	Start the Local Test Environment Docker images. Use the **--database-max-heap-size** parameter to set the size of the Local Test Environment database, in bytes; for example: **linux-apic-lte start --database-max-heap-size 4096M** **linux-apic-lte start --database-max-heap-size 1G** **linux-apic-lte start --database-max-heap-size 1048576K** **linux-apic-lte start --database-max-heap-size 1073741824** The default value is **1024M**. Tip: By default, the **platform-apic-lte start** command deletes all previous data and re-initializes the Local Test Environment configuration, so all your previous configuration, including published Products, is deleted. To retain the previous configuration, and to apply the same command parameters that were used in the previous **platform-apic-lte start** command, supply the **--keep-config** parameter.
platform-apic-lte status	Display status information for the Local Test Environment components, and endpoint and authentication details.
platform-apic-lte stop	Stop the Local Test Environment Docker images.
platform-apic-lte version	Display Local Test Environment version information.

Troubleshooting the Local Test Environment

You can consult the log file for each Local Test Environment microservice or database by using the following command:

```
docker logs container-name
```

where *container-name* is one of the following:

- **apic-lte-juhu**: the authentication gateway
- **apic-lte-apim**: the API Management service

- `apic-lte-lur`: the Local User Registry
- `apic-lte-db`: the Postgres database of the API Management service
- `apic-lte-datapower-api-gateway`: the DataPower API Gateway
- `apic-lte-datapower-gateway`: the DataPower Gateway (v5 compatible)

You can access the gateway logs in either of the following ways:

- Use the gateway administration web UI:
  1. Open the page `https://localhost:web_ui_port` in a browser; for details of the required port value, see Local Test Environment port values.
  2. Select the apiconnect domain and the WebGUI interface, and log in with user name `admin` and password `admin`.
  3. Click View Logs.
- Use the gateway administration CLI:
  1. Open an SSH connection by using the following command:

     ```
     ssh -p gateway-ssh-port localhost
     ```

     For details of the required port value, see Local Test Environment port values. The user name is `admin` and the password is `admin`.
  2. Enter the command **switch domain apiconnect**.
  3. To view the gateway log, enter the command **show log**.
  4. To view the log for the communication between the gateway and the API management system, enter the command **show logging gwd-log**.

Local Test Environment port values

If any of the default port values for the Local Test Environment components conflict with ports already in use on your system, you can change them when you start the Local Test Environment by passing one or more `--component port_value` parameters to the `platform-apic-lte start` command, where:

- *component* is the Local Test Environment component whose port value you want to change.
- *port_value* is the required value.

For example:

```
platform-apic-lte start --datapower-api-gateway-api-port 9445
```

The following table lists the components, together with the corresponding *component* parameters, and the default port values:

Component	*component* parameter	Default port value
DataPower API Gateway API port	`datapower-api-gateway-api-port`	9444
DataPower API Gateway API Connect service port	`datapower-api-gateway-apic-service-port`	3001
DataPower API Gateway REST management port	`datapower-api-gateway-rest-management-port`	5555
DataPower API Gateway SSH port	`datapower-api-gateway-ssh-port`	9023
DataPower API Gateway administration web UI	`datapower-api-gateway-web-gui-port`	9091
DataPower API Gateway XML management port	`datapower-api-gateway-xml-management-port`	5551
DataPower Gateway (v5 compatible) API port	`datapower-gateway-api-port`	9443
DataPower Gateway (v5 compatible) API Connect service port	`datapower-gateway-apic-service-port`	3000
DataPower Gateway (v5 compatible) REST management port	`datapower-gateway-rest-management-port`	5554
DataPower Gateway (v5 compatible) SSH port	`datapower-gateway-ssh-port`	9022
DataPower Gateway (v5 compatible) administration web UI	`datapower-gateway-web-gui-port`	9090
DataPower Gateway (v5 compatible) XML management port	`datapower-gateway-xml-management-port`	5550
Platform API port	`platform-api-port`	2000

Related information

- Creating TLS Client Profile in the Local Test Environment

Testing an API with the Explorer tab

Use the Explorer tab in the API editor to see how your APIs look to a consumer in the Developer Portal. You can also use the Try it tool to test the behavior of the API.

Before you begin

To test an API in the Try it test tool, the API must be auto-published. If the API is not auto-published, the Try it tool isn't enabled in the Explorer.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

About this task

The Explorer tab in the API editor of the API Designer UI and API Manager UI enables you to view how your APIs look to an API consumer in the Developer Portal. You can check the descriptions of the different artifacts, and as well as any schemas or examples, and you can also test the behavior of the API by using the Try it tool.

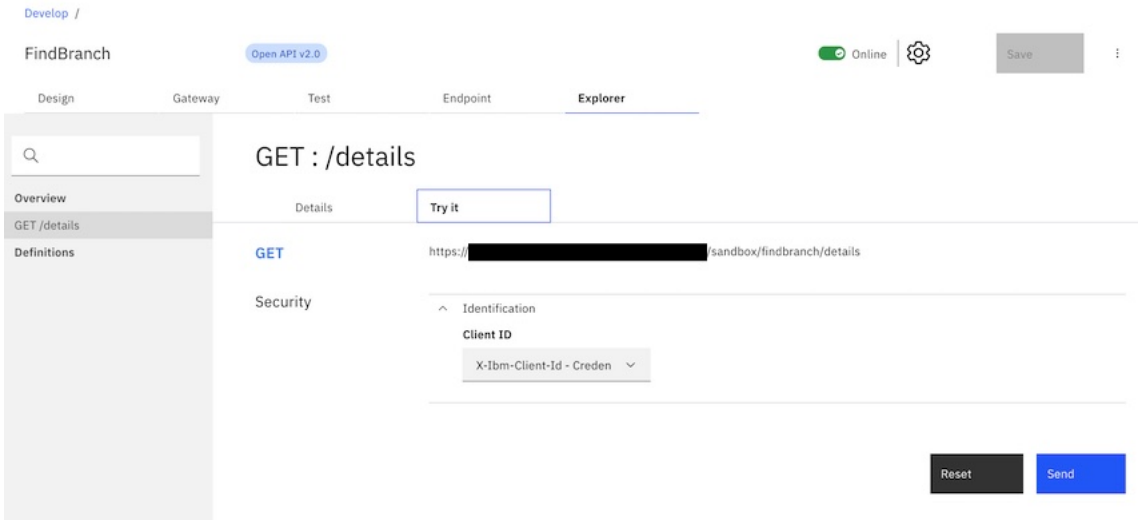
Note:

- You cannot use the Try it tool to test non-enforced APIs.

Procedure

To test an API by using the Try it tool, complete the following steps.

1. If you are using API Designer, set the mode to Online using the Options menu ⋮ on the main page.
2. In the navigation pane, click Develop, then select the APIs tab.
3. Click the title of the API that you want to work with.
   The API editor is displayed.
4. Click the Explorer tab.
   An Overview of the API is displayed.
5. Click the API artifacts in the side menu bar to explore the operations and definitions that are available in the API, including reviewing any examples and schemas.
6. To test API behavior, click an operation in the side menu bar, and click the Try it tab.
   The test pane for the operation is displayed, as shown in the following example:



7. Provide any required API parameters, and click Send.
   The API response is displayed in the Response section.
   Note: If you receive a message relating to an untrusted certificate, click the link provided, accept the certificate, then return to the test environment and click Send again.

What to do next

When you are happy with the look and behavior of your API, you can stage the API to a Product in a Catalog. For more information, see Staging an API.

Staging an API

The graphical wizard provides an option that adds the API to a Product and stages the Product in a Catalog. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.

Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see Creating and configuring Catalogs.

Ensure that the Catalog has at least one gateway service configured.

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in IBM API Connect®.
To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Stage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see Creating and configuring Catalogs or Managing user access in a Space.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Staging is not available when working offline in API Designer.

A Catalog is a staging target, and behaves as a logical partition of the DataPower® Gateway, and the Developer Portal.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (https://www.npmjs.com/package/@apidevtools/swagger-parser).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either **`X-IBM-Client-Id`** or **`X-IBM-Client-Secret`**.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.

- De-referencing of local references in the definition file (that is, values of **`$ref`** properties), and ensuring these are valid JSON Pointers within the file.

Note: If the OpenAPI file that defines your API uses a **`$ref`** field to reference a fragment of OpenAPI code that is defined in a separate file, the **`$ref`** field is replaced with the contents of the target file before the product that contains the API is staged or published (the **`$ref`** field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
3. You can stage an API either from the Develop listing page, or from within the API definition itself.
   a. To stage an API from the Develop listing page, click the options menu icon alongside the required API, and then select Stage.
   b. To stage an API from within the API definition, complete the following steps:
      i. Click the title of the API that you want to work with.
      ii. Click the options menu icon:

      iii. Click Stage.
4. Choose one of the following actions:
   - To stage the API by adding it to a new Product:
     - Select New Product - Publish using a new product.
     - When prompted, enter a Title and Version.
     - The Product Name is automatically entered. The Name is the used to refer to the product in CLI commands. See apic products.
     - Click Next.
   - To stage the API by adding it to an existing Product:
     - Select Existing Product - Publish using an existing product
     - Select the Product you want to use.
     - Click Next.
5. On the Stage To page, select the Catalog to which you want to stage the Product.
   Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to.
   If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

   For details of how to create a Catalog in a provider organization, see Creating and configuring Catalogs.
6. If, when the staged Product is subsequently published, you want it to be published only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see API Connect gateway types.
7. Click Stage.

Results

Your Product is staged to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you staged the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Staged.

For information about the lifecycle of a product, see The Product lifecycle.

If approval is required to stage Products in the Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is staged when the request is approved. If approval is not required, the Product is staged immediately.

For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

Publishing an API

The user interface provides an option that adds an API to a Product and publishes the Product in a Catalog. Publishing a draft Product makes the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.

Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see Creating and configuring Catalogs.

Ensure that the Catalog has at least one gateway service configured.

Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in IBM API Connect®.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Manage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see Creating and configuring Catalogs or Managing user access in a Space.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Publishing is not available when working offline in API Designer.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (https://www.npmjs.com/package/@apidevtools/swagger-parser).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either `X-IBM-Client-Id` or `X-IBM-Client-Secret`.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
  - De-referencing of local references in the definition file (that is, values of `$ref` properties), and ensuring these are valid JSON Pointers within the file.

Note: Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "`(?`" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
    definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the product that contains the API is staged or published (the `$ref` field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
3. You can publish an API either from the Develop listing page, or from within the API definition itself.
   a. To publish an API from the Develop listing page, click the options menu icon ⋮ alongside the required API, and then select Publish.
   b. To publish an API from withing the API definition, complete the following steps:
      i. Click the title of the API that you want to work with.
      ii. Click the options menu icon:

      iii. Click Publish.
4. Choose one of the following actions:
   - To publish the API by adding it to a new Product:
     - Select New Product - Publish using a new product.
     - When prompted, enter a Title and Version.
     - The Product Name is automatically entered. The Name is the used to refer to the product in CLI commands. See apic products.
     - Click Next.
   - To publish the API by adding it to an existing Product:
     - Select Existing Product - Publish using an existing product
     - Select the Product you want to use.
     - Click Next.
5. Select the Catalog to which you want to publish the Product.
6. If Spaces have been enabled in the selected Catalog, select the Space that you require.
   Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to.
   If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

   For details of how to create a Catalog in a provider organization, see Creating and configuring Catalogs.

7. If you want to publish the Product only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see API Connect gateway types.
8. Click Publish.

Results

Your Product is published to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you published the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Published.

For information about the lifecycle of a product, see The Product lifecycle.

If approval is required to publish Products in the Catalog, a publish approval request is sent, and the Product moves to the Pending state; the Product is published in the Catalog when the request is approved. If approval is not required, the Product is published immediately.

Note: If approval is required to stage Products to the Catalog, a stage approval request is sent. When the request is approved, the Product moves to the staged state and must be separately published in the Catalog. For information on publishing a staged Product in a Catalog, see Publishing a Product.
For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

Notes:

- If an OpenAPI 3 API contains a response wildcard (which is not supported), publishing is disabled for that API. You must correct the problem before you can publish the API.
- If the Product contains an API that includes a user-defined policy that has not been imported into this Catalog, the publish will fail. The user-defined policy must be available in the required Catalog for the Product to be published successfully. For detailed instructions on how to import a user-defined policy into a Catalog, see Importing a user-defined policy into a Catalog.

Creating a new version of an API definition

You can create multiple versions of an API definition and edit the versions independently by using IBM® API Connect.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

Note: If you want to publish a modified API to a production Catalog, you **must** create a new version of the API. You cannot publish an API to a production Catalog if there is already a published API with the same name and version.

Procedure

To create a new version of an API definition, complete the following instructions:

1. In the navigation pane, click Develop, then select the APIs tab.
2. You can create a new version of an API definition either from the Develop page, or from within the API definition itself.
   a. To create a new version of an API definition from the Develop page, complete the following steps:
      i. Alongside the API version that you want to work with, click the options icon ⋮, then click Save as a new version. The Save API as a new version window opens.
      ii. Enter your new version number, then click Save.
   b. To create a new version of an API definition from within the API definition itself, complete the following steps:
      i. Click the title of the API definition that you want to create a new version of.
      ii. Click the options menu icon:

      iii. Click New Version, enter your new version number, then click Submit.

   Note: The version corresponds to the **`info.version`** property value of the API's OpenAPI definition. The **`version.release.modification`** version numbering scheme is recommended, for example `1.0.0`.

Results

You have created a new version of your API definition, which you can now edit independently of other versions. Each version of the API definition is listed separately on the APIs and Products page.

Related tasks

- Creating an API definition

Updating an API definition from a file

You can update an API by uploading a .yaml or .json file that contains the OpenAPI definition for the revised API.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

The update operation completely replaces the previous API definition.

Procedure

To update an API from a file, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
2. Click the title of the API that you want to update.
3. Click the options menu icon:

4. Click Update, then either drag and drop the file or click the link to select the file from your local file system.
5. Click Submit, then click Save.

Updating a SOAP API

You can update the configuration of an existing SOAP API either by uploading a WSDL file, or by uploading a .zip archive that contains a primary WSDL file together with other WSDL or XSD files that it references.

About this task

You complete this task by using the browser based API Manager UI.

You can also use the API Connect developer toolkit CLI to update a SOAP API; for more information see API development and management commands.

When you initiate the update action, the specified WSDL is parsed to find a service that matches the name of the API that you are updating. If a match is found, the WSDL service is converted into a YAML file that is used to update the API. If no match is found, an error message is displayed.

Only those sections of the API that are affected by the new WSDL are replaced, the other sections are unchanged.

Procedure

To update a SOAP API from a WSDL file or a .zip archive, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
2. Click the title of the SOAP API that you want to update.
3. Click the options menu icon:

4. Then click Update WSDL, and either drag and drop the file or click the link to select the file from your local file system.
   The format of the WSDL file is validated. If the format is invalid, an error message is displayed.
5. Click Next.
   The services in the selected API are displayed. If no match is found, an error message is displayed.
6. Click Upload.
   The SOAP API is updated from the specified WSDL definition.
7. Click Save to save the updates.
8. If you updated a previously published API, you must publish it again for the change to take effect:
   a. Click the Online/Offline toggle (shown in step 3) to set the API as Offline.
   b. Click the toggle again to set the API back to Online, which publishes the API again.

Related tasks

- Creating an API definition

Renaming an API definition

By renaming an API definition, you change the **`name`** property that, together with the version, is used to uniquely identify the API definition in API Connect. The **`name`** property is initially generated automatically when you first create the API definition, based on the title, but you can later change it if desired.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.

2. Click the title of the API that you want to work with.
3. Click the options menu icon:

4. To rename the API, select Rename.
   Only this version of the API is renamed.
5. Enter the new name, then click Submit.
   Note: If you previously added the API to a Product, it will no longer be recognized by that Product. You must open the Product and modify the `apis:` section on the Source tab as appropriate; for example:

```
apis:
  newname1.0.0:
    name: 'newname:1.0.0'
```

Changing an API rate limit

When you activate an API to make it available for testing, as described in Activating an API, no rate limit is applied to restrict calls to the API. However, if desired, you can apply a rate limit to the API, or modify a previously applied rate limit. The rate limit defines the maximum number of calls allowed in a specified time period.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Note: For information about rate limits and burst limits in API Connect, see Understanding rate limits for APIs and Plans.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Click the options menu icon:

4. Click Edit Rate Limit.
5. Select Unimited to allow unrestricted calls to the API, or select Custom Rate Limit to define the required rate limit; use the supplied fields to define the maximum number of calls allowed in a specified time period; for example, 100 calls per 1 minute.
6. Click Submit when done.

Adding an OpenAPI extension to an API

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An OpenAPI extension is imported into a Catalog, or to a Space within a Catalog, then added to the API schema.

Before you begin

Import the OpenAPI extension into the Sandbox Catalog, or into any Catalog to which the API that you are adding the extension to will be published; see Importing an OpenAPI extension into a Catalog.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Api-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
2. Click the title of the API that you want to work with.
3. Select the Gateway tab, expand Gateway and portal settings, then click Extensions.
4. Select the OpenAPI extensions that you want to use with this API, then click Save.
   Note: The OpenAPI extensions listed are those that have been imported into the Catalog and gateway service defined in the testing preferences for the API; see Importing an OpenAPI extension into a Catalog and Specifying the testing preferences for an API. You will also need to import the OpenAPI extension into any Catalog to which the API will be published.

5. Update the OpenAPI source for the API definition with the values for the fields as defined in your selected OpenAPI extension schema files, as follows:
   a. Click the Source icon </>.
   b. Add a property whose name must begin with **x-** and references the name of the OpenAPI extension that you created. Specify the values of the properties that were defined in the OpenAPI extension YAML file.
      For an example, see Referring to an extension in an API definition
   c. Click Save when done.

Downloading an API definition

You can download the details of your API definitions so that you can store them for later recovery.

Before you begin

To complete this task, you must have created an API definition. For more information, see Creating an API definition.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **Drafts:View** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

You can download REST APIs that were created by using Creating a REST proxy API from a target service for example. This download creates a .yaml file that defines the API.

You can download SOAP APIs that were created by using Creating a REST proxy API from an existing WSDL service , or Creating a SOAP proxy API from an existing WSDL service for example. This download creates a .zip file that contains the WSDL definition and the YAML file that defines the API.

You can use the downloaded .yaml and .zip files to re-create your API. For information about creating an API from a .yaml file, see Adding a REST API by importing an OpenAPI definition file. For information about creating an API from a .zip file, see Adding a SOAP API by importing a zip file.

Procedure

To download a .yaml, or .zip file that contains the details of your API, complete the following steps:

1. In the navigation pane, click Develop, then select the APIs tab.
2. Alongside the API version that you want to work with, click the options icon and then click Download.
3. Save the file to the required location.

Results

You downloaded a .yaml, or .zip file with a representation of your API.

Deleting an API definition

You can delete an API definition that is no longer required.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **Api-Drafts:Edit** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
2. You can delete an API either from the Develop listing page, or from within the API definition itself.
   a. To delete an API from the Develop listing page, click the options menu icon alongside the required API, and then select Delete.
   b. To delete an API from within the API definition, complete the following steps:
      i. Click the title of the API definition that you want to work with.
      ii. Click the options menu icon:

      iii. To delete just this version of the API, select Delete. To delete all versions of this API, select Delete API.
   Note: If a Product was previously generated automatically from the API, by using the Activate API option, that Product will be deleted together with the API; such Products have the title *api_title* auto product.
3. Click Delete to confirm.

Using an options file when importing a WSDL service

When you create an API definition, or add a target WSDL service to an API definition, by importing a .zip file, you can specify additional directives by including an options file in the .zip file.

You can use an options file when importing a .zip file while performing any of the following tasks:

- Creating a REST proxy API; see Creating a REST proxy API from an existing WSDL service.
- Creating a SOAP proxy API; see Creating a SOAP proxy API from an existing WSDL service.
- Adding a target WSDL service to an API definition; see Editing an API definition.

The options file is a YAML file, and must have the file name apiconnect.yaml.

You can include the following fields in the file:

Table 1. Fields that can be included in the apiconnect.yaml file

Field name	Value	Default	Description
`suppressExamples`	`true` or `false`	`false`	Suppress auto-generation of examples.
`wssecurity`	`true` or `false`	`true`	Enable generation of definitions for WS-Security headers.
`implicitHeaderFiles`	Array of XSD file locations		Additional schema files, not referenced in the WSDL, that are used to define additional SOAP headers.
`port`	The `name` attribute for a `wsdl:port` in a `wsdl:service` definition.		Create the API by using the information for the specified port. If the specified port does not exist, or refers to a REST/XML port rather than a SOAP port, the API creation fails. If no `port` field is present in the options file, API Connect creates the API by using the first SOAP port that it finds in the `wsdl:service` definition.

Example

```
suppressExamples: true
wssecurity: false
implicitHeaderFiles:
  - xsdDir/schema1.xsd
  - xsdDir/schema2.xsd
```

Variable references in API Connect

In API Connect you can reference different variables in your API definition.

When defining an API, creating a custom policy, or configuring another policy or logic construct, you can include references to context variables and properties.

Variable references are resolved either when the API is staged in a Product, for static variables that are fixed upon staging, or when the API is called, for variables that can change with each API call.

Types of variables

Context variables
: A context variable is a variable relevant during an API call, for example, the input of the call, the path of the call, or the message during the call. A context variable is one of the variables that makes up that particular context.
Context variables can consist of more than one part, for example, `request.headers`.

  For a list of available context variables, see API Connect context variables.

API properties
: An API property is a variable in an API where its value depends upon the Catalog in which the API is staged or published. By referencing an API property, you can use the same API definition in different Catalogs where there are small differences between the instances of the API between the Catalogs. For example, an assembly could contain an `if` construct that executes its case when a particular Catalog is used, determined from the value of the API property. API properties can also be used to hide a value such as a password by encoding the value.
API properties are referenced by name.

  For a list of API properties, see API properties

  For more information, see Setting API properties (OpenAPI 2.0) or Setting API properties (OpenAPI 3.0).

  Note: Once defined, an API property is read only.

Catalog properties
: A Catalog property is specific to a Catalog. and can be referenced in any of the API definitions in that Catalog. For more information, see Creating and configuring Catalogs.

Notes:

- If you change the value of a Catalog property, any API that references that property must be republished for it use the new value.
- Catalog properties and API properties are not supported with global policies. Therefore, if you use such properties in a global policy, they are not replaced with the values specified in the Catalog or API property definitions.

Methods of referencing variables

You can get, or set, the value for a referenced variable.

- Get the value for a variable in either of the following ways:
  - Through the GatewayScript policy, run the `apim.getvariable()` API.
  - API Gateway only Through the GatewayScript policy, use the `context.get()` function; for more information, see Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.
  - Through the XSLT policy, run a stylesheet that uses the `apim:getVariable` extension function.
  - In an assembly policy field that supports variable references, use the following syntax:

    `$(variable)`

- Set the value for a variable in either of the following ways:
  - Through the Set Variable policy.
  - Through the GatewayScript policy, use the `apim.setvariable()` API.
  - API Gateway only Through the GatewayScript policy, use the `context.set()` function; for more information, see Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.
  - Through the XSLT policy, run a stylesheet that uses the `apim:setVariable` extension element.

GatewayScript references

When you want to reference a variable in a GatewayScript context, use one of the following methods:

`apim.getvariable(variable)`

where *variable* is the name of the context variable or API property that you want to reference.

`apim.setvariable(variable, value, action)`

where

- *variable* is the name of the context variable or API property that you want to reference.
- *value* is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, use the following code:

  ```
  var contentType = apim.getvariable('request.headers.content-type');
  apim.setvariable(variable, contentType, 'set');
  ```

  This property is required only when `set` or `add` is specified as the action.
- *action* is the action that you want to apply to the variable. Valid options are:
  - `set`
  - `add`
  - `clear`

  If no option is set, the default option of `set` is applied.

Use the `getvariable` method to retrieve the value of a context variable or API property, and the `setvariable` method to change one.

Some of the situations where you would use this type of reference are:

- GatewayScript policies. -For more information, see GatewayScript.
- `if` logic constructs. For more information, see if.
- User-defined policies. For more information, see Authoring policies.

Stylesheet references

You can reference a variable by using functions and elements in an XSLT policy with the following syntax:

```
<xsl:variable name="variable_name" select="apim:getVariable(variable)" />
```

where *variable* is a literal value, another variable, or a valid XSLT XPath statement.

```
<xsl:call-template name="apim:setVariable">
    <xsl:with-param name="varName" select="variable"/>
    <xsl:with-param name="value" select="value"/>
    <xsl:with-param name="action" select="action"/>
</xsl:call-template>
```

where

- *variable* is the name of the context variable or API property that you want to reference. This can be a literal value, another variable, or a valid XSLT XPath statement.
- *value* is the string value that you want to set the variable to. This can be a literal value, another variable, or a valid XSLT XPath statement. This property is required only when `set` or `add` is specified as the action.
- *action* is the action that you want to apply to the variable. This can be a literal value, another variable, or a valid XSLT XPath statement. Valid options are:
  - `set`
  - `add`
  - `clear`

  If no option is set, the default option of `set` is applied.

The following example sets a named variable to the value of the Content-Type header in a request:

```
<xsl:variable name="contentType" select="apim:getVariable('request.headers.content-type')" />
<xsl:call-template name="apim:setVariable">
    <xsl:with-param name="varName" select="'variable'"/>
    <xsl:with-param name="value" select="$contentType"/>
    <xsl:with-param name="action" select="'set'"/>
</xsl:call-template>
```

Inline references

In many situations you can make a simpler reference, by using the following syntax:

```
$(variable)
```

where *variable* is the name of the context variable or API property that you want to reference.
Some of the situations where you would use this type of reference are:

- The URL called by an Invoke or Proxy policy. For more information about the policies, see Invoke or Proxy.
- A Map policy. For more information, see Map.

Notes: The map policy can reference the following variables inline:

- Variables that are defined as inputs to the map policy and specified in the from field of a mapping.
- Context variable or API properties, provided that the **x-ibm-gateway-map-resolve-apic-variables** API property is not set to **false**. If an inline reference in a map policy resolves to a context variable or API property, it is immediately replaced by the corresponding value. For more information on the **x-ibm-gateway-map-resolve-apic-variables** API property, see Properties controlling the map policy.

For more information on the use of inline references, and examples, see the following About inline references section.

About inline references

When you use an inline reference for a property value in an assembly policy, the variable reference is replaced with the corresponding string value at run time, before the property is evaluated. For example, consider the following incoming request, which has two headers and two query parameters (a **curl** command is used to illustrate the request):

```
curl --data-binary @request.json \
   -H 'Content-Type: application/json' \
   -H 'X-Environment: test' \
   http://apiserver/org/catalog/petstore/getPetDetails?petid=285&storeid=z03
```

If an **invoke** policy is configured as follows:

```
invoke:
   target-url: https://backend/GetPetInfo/$(request.parameters.storeid)/$(request.parameters.petid)
```

then the value of the **target-url** property is evaluated as **https://backend/GetPetInfo/z03/285**, and this is therefore the URL to which the **invoke** policy sends its request.

DataPower Gateway (v5 compatible) only You can also use an inline reference in the **condition** property of a **switch** policy. Consider the following example:

```
- switch:
    case:
        - condition: "$(request.headers.X-Environment)" == "production"
          execute:
              - invoke:
                  http://www.finance.com/base/stockquote
        - condition: "$(request.headers.X-Environment)" == "test"
          execute:
              - invoke:
                  http://testserver1.finance.com/stockquote
```

Using the incoming request referred to previously, the value of the **X-Environment** header is **"test"**. The inline references are first evaluated, then the conditions are evaluated as JavaScript. Therefore, the first condition is evaluated as **"test" == "production"**, returning **false**, and the second condition is evaluated as **"test" == "test"**, returning **true**, so the **invoke** policy sends its request to **http://testserver1.finance.com/stockquote**.
Important: This example, which encloses the inline references inside double quotes, assumes that the API is deployed to the DataPower API Gateway. If you deploy the API to the DataPower Gateway (v5 compatible), then the double quotes are added implicitly when the inline references are evaluated, and you must therefore omit them from the **condition** property; the two conditions in this example would therefore be

```
condition: $(request.headers.X-Environment) == "production"
```

and

```
condition: $(request.headers.X-Environment) == "test"
```

However, as an alternative to using an inline reference in a **condition**, you can instead use pure JavaScript, as follows:

```
- switch:
    case:
        - condition: request.headers["X-Environment"] == "production"
          execute:
              - invoke:
                  http://www.finance.com/base/stockquote
        - condition: request.headers["X-Environment"] == "test"
          execute:
              - invoke:
                  http://testserver1.finance.com/stockquote
```

in which case the policy configuration will work correctly as-is in both types of gateway.
For more information on the types of gateway, see API Connect gateway types.

In addition to single strings, you can use an inline reference for a complete object. Consider the following example, which uses an inline reference in a **set-variable** policy:

```
- set-variable:
    actions:
        - set: saved.reqHdrs
          value: $(request.headers)
          type: string
```

Here, the **request.headers** structured JSON object is substituted with its corresponding string representation, and a variable named **reqHdrs** is therefore created, under the **saved** object, with a string value of **{"Content-Type":"application/json","X-Environment":"test",...}**. However, if you are using the DataPower API Gateway, you can specify **type: any**, in which case the **set-variable** policy copies the value as-is, and **saved.reqHdrs** is created as a JSON object rather than a string:

```
- set-variable:
    actions:
        - set: saved.reqHdrs
          value: $(request.headers)
          type: any
```

You can reference a property in a JSON or XML request directly by name, by using the following syntax:

$(request.body.*property_name*)

For example, if the request contains **{ "account-number": 123 }** or **<account-number>123</account-number>** you can retrieve the value **123** by using the following inline reference:

$(request.body.account-number)

If the property value is within a nested property structure, you can reference it by concatenating the property names. For example, if the request contains **{ "account": { "balance": 123 } }** or **<account><balance>123</balance></account>** you can retrieve the value **123** by using the following inline reference:

$(request.body.account.balance)

Similarly, you can reference a property in a JSON or XML response by using the following syntax:

$(message.body.*property_name*)

Restriction: If you are using the DataPower API Gateway, you can reference properties in this way only if the format of the request or response is JSON; the mechanism is not supported with XML request or response formats. If the you are using the DataPower Gateway (v5 compatible) the mechanism is supported with both JSON and XML formats.

API properties

Properties are used by the gateway to control behavior of certain policies. Typically, you provide properties, but the policy can also provide properties settings.

In IBM® API Connect, you can create API properties that consist of Catalog-specific values to eliminate the need for source code modifications. You can then reference the properties elsewhere in your API definition.

Pre-supplied API properties for various policies are shown in the tables.

A list of invoke related API properties that control the behavior of the invoke policy.

Table 1. Properties controlling the invoke policy

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-decode-request-params	No	If set to a value of **true**, any request parameters that are referenced by a variable definition on an invoke target-url are URL-decoded. The default behavior is to not decode any parameters, thereby sending them to the target URL without alteration.	Boolean

DataPower Gateway (v5 compatible) only
x-ibm-gateway-invoke-suppress-clientid
	No	When set with a value of **true**, or not specified, the X-IBM-Client-Id HTTP header (if specified on the API request) is suppressed from being sent to the invoke target URL. When set with a value of **false**, the X-IBM-Client-Id HTTP header is no longer suppressed from being sent to the invoke target URL.
		This property is supported only by the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway then to achieve the same functionality add a **header-control** property to the **invoke** policy configuration in the OpenAPI definition for your API, as in the following examples.
		
		Suppress the **X-Client-ID** header as follows:
		`- invoke:`
		`  target-url: http://myhostname/mypath`
		`  header-control:`
		`  type: blacklist`
		`  values:`
		`  - ^X-Client-ID$`
		
		Suppress the **userId** query parameter as follows:
		`- invoke:`
		`  target-url: http://myhostname/mypath`
		`  parameter-control:`
		`  type: blacklist`
		`  values:`
		`  - ^userId$`
			Boolean

DataPower Gateway (v5 compatible) only x-ibm-gateway-optimize-invoke	No	If set to `false`, prevents the replacement of the last invoke in a policy with proxy. Any value other than `false` (case insensitive) will result in the last invoke in a policy possibly being replaced by proxy when the API is executed in the gateway.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-queryparam-encode-plus-char	No	If set to a value of **true**, all "+" characters in the query parameter values of the target-url of the Invoke and Proxy policies are encoded to "%2F". The default value is **false**.	Boolean

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-api-enforce-response-limits	No	If set to a value of **true**, allows the JSON parser to be enforced on the response rule. If the response body size is higher than the JSON parser limit set in the DataPower domain, a status code of 500 is returned. Note: The x-ibm-gateway-api-enforce-response-limits property is supported by the DataPower Gateway (v5 compatible) but not by the DataPower API Gateway. However, if you are using the DataPower API Gateway, consider using a Parse policy in your API assembly to enforce these limits. For information on the different types of gateway, see API Connect gateway types.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-invoke-emulate-v4-soap-error	No	IBM API Management Version 4.0 initiates a DataPower error when a SOAP fault is returned from a web service. IBM API Connect provides a mechanism to catch SOAP errors and does not initiate a DataPower error. For compatibility with APIs developed in IBM API Management Version 4.0, set this property to **true** only in the case where a gateway extension is expecting to handle a SOAP error in a post error rule. The default value is **false**. Note: This property is deprecated in favor of x-ibm-gateway-invoke-emulate-v4-invoke-error.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-invoke-keep-payload	No	If set to a value of **true**, the invoke policy sends a payload on an HTTP DELETE method. This property is available for use with IBM DataPower Gateway version 7.7.1.1 and later. The default value is **false**.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-invoke-emulate-v4-invoke-error	No	IBM API Management Version 4.0 initiates a DataPower error when a backend server error is returned, either a SOAP fault returned from a web service or a JSON or XML (non-SOAP) error from a restful service. IBM API Connect provides a mechanism to catch SOAP errors and operation errors and does not initiate a DataPower error when they occur. If no catch policy is configured, a generic error message is generated. For compatibility with APIs developed in IBM API Management Version 4.0, set this property to **true** only in the case where a gateway extension is expecting to handle a backend server error in a gateway extension post error rule or if the client of the API is expecting the backend server error to be returned. The default value is **false**.	Boolean

A list of map-related API properties that control the behavior of the map policy.

Table 2. Properties controlling the map policy

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-array-first-element-value	No	In IBM API Management Version 4.0, if a mapping source value is from an array then only the first value is output. In API Connect, the default behavior is to return an array of all array element values. To maintain compatibility with IBM API Management Version 4.0, set this API property to **true** to only return the first array element value.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-resolve-apic-variables	No	By default, any API Connect variable that is found in the map configuration is resolved. For example, **$(request.headers.content-type)** resolves to the request's content type header. Because searching for variables in every map property can be CPU intensive, you can choose not to resolve variables by setting this API property to `false`. If this property is not configured or is set to any other value, the existing behavior to search for these variables continues. Note that variable usage within a map value JavaScript snippet is not changed provided that the variables that are referenced come from a configured map input.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-create-empty-array	No	This property controls how the map policy handles the output of an empty array; it can have the following values: • **all**: Output all empty arrays, including empty children arrays. This is the default value if the property is not configured or has an invalid value. • **parent**: Output only the current property's empty array value. Children map actions of this property are not attempted. • **none**: Prevent any empty output array values from being produced.	String
DataPower Gateway (v5 compatible) only x-ibm-gateway-optimize-schema-definition	No	Set the value of this property to **true** to provide a performance improvement to the map policy when a very complex schema definition is referenced by a policy output definition; for example, some very complex schemas that are generated by importing a very complex WSDL schema. The map policy builds a schema from an API definition when a referenced definition is provided as the value of the schema. If the schema does not have references that generate a circular reference, setting this property to **true** might provide a performance benefit while generating the same schema as would otherwise have been generated. However, in cases where the schema is very complex, with many potentially circular references, the generated schema could be different because the enhanced schema handling processes circular references differently. In such cases therefore, you should examine the resulting output to determine if the performance benefit gained is not at the expense of a change in the map policy output. The default value of this property, is **false**, maintaining the existing behavior and performance.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-null-value	No	Set the value of this API property to **true** to allow a property from a map policy's input data with a value of **null** to be mapped to the output document. By default, a property from a map policy's input data with a value of **null** is not mapped to the output document.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-resolve-xmlinput-datatypes	No	XML input elements with numeric or boolean data have no metadata to indicate whether this data should be mapped as a string value or as the specific data type. If you set the value of this property to **false**, XML input elements are always mapped as a string. If you set the value to **true**, numeric or boolean XML input elements are mapped as the corresponding data type from the input schema. The default value is **false**.	Boolean

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-map-xml-empty-element	No	This property controls how the map policy handles XML input empty elements and impacts JSON output when the input document is XML; it can have the following values: • **`string`**: The value for an empty XML element is considered to be an empty string. This is the default value if the property is not configured or has an invalid value. • **`null`**: The value for an empty XML element is considered to be null. A mapping of this element to a JSON output property does not occur unless the API property x-ibm-gateway-map-null-value is also specified with a value of **`true`**. • **`none`**: The empty XML element is ignored. • **`string-badgerfish`**: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property. • **`null-badgerfish`**: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the API property **`x-ibm-gateway-map-null-value`** is also specified with a value of **`true`**.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-schema-definition-reference-limit	No	Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5. If you specify a value greater than 5, a value of 5 is assumed. If you specify a non-numeric value, a value of 1 is assumed.	String

DataPower Gateway (v5 compatible) only
x-ibm-gateway-map-emulate-v4-default-required-properties
	No	Set this property to true to have default values generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:
		• An array consists of objects that contain one or more required properties.
		• An object which is optional has one or more child properties that are required.
		By default, these required properties are not present in the output. If you set the **`x-ibm-gateway-map-emulate-v4-default-required-properties`** API property to **`true`**, these required properties will be present in the output. If the output schema defines a **`default`** property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:
		• String: empty string ("")
		• Number: 0
		• Boolean: false
		• Object: empty object
		• Array: empty array
		Example 1
		The input data has the following array of objects:
		`[{"a": "value1"}, {"a": "value2", "b": "value3"}]`
		The output schema defines the output object as having two properties, **`a`** and **`b`**, of which **`b`** is required. The map policy defines the following mappings:
		• **`input.array.a`** to **`output.array.a`**
		• **`input.array.b`** to **`output.array.b`**
		If the **`x-ibm-gateway-map-emulate-v4-default-required-properties`** API property is set to **`true`**, and **`b`** is either not mapped or has no input data present, then **`b`** is assigned a default value of an empty string, and the output is as follows:
		`[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]`
		Example 2
		The output schema defines the following structure:
		`{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }`
		Property **`b`** is optional but property **`d`** within **`b`** is required.
		The map policy defines a mapping to **`output.a.b.c`**.
		If the **`x-ibm-gateway-map-emulate-v4-default-required-properties`** API property is set to **`true`**, and **`d`** is not mapped, then **`d`** is assigned a default value of an empty string, and the output is as follows:
		`{"a" : {"b" : {"c" : "value1", "d" : ""} } }`
		If the **`x-ibm-gateway-map-emulate-v4-default-required-properties`** API property is not specified or does not have a value of **`true`**, these required properties are **not** created in the output with their default values.
			Boolean

DataPower Gateway (v5 compatible) only x-ibm-gateway-map-post-process-json-output	No	Set the value of this property to **`true`** to enable post processing of mapped JSON output. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. Set the value to **`false`** for no post processing of mapped JSON output. The default value is **`false`**.	Boolean

Property	Required	Description	Data type

DataPower Gateway (v5 compatible) only
x-ibm-gateway-map-emulate-v4-empty-json-object
	No	If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Set the value of this property to **true** to create an empty object for the parent of the target mapping, emulating the behavior of IBM API Management Version 4.0.
		
		Example
		The map policy defines a mapping to **output.a.b.c**.
		If input data is present, the output is as follows:
		
		`{ "a": { "b": { "c": "inputvalue" } } }`
		
		If there is no input data, and the **x-ibm-gateway-map-emulate-v4-empty-json-object** API property is set to **true**, the output is as follows:
		
		`{ "a": { "b": { } } }`
		
		Properties **a** and **b** are created but the value of **b** is an empty object.
		The default value is **false**.
			Boolean

DataPower Gateway (v5 compatible) only A list of proxy-related API properties that control the behavior of the proxy policy.

Table 3. Properties controlling the proxy policy

Property	Required	Description	Data type

x-ibm-gateway-proxy-suppress-clientid	No	A setting of `false` activates the injection of the X-IBM-Client-Id HTTP header (if it is specified on the API request), or the **client_id** query parameter in the request URL, to the proxy target-url. If not specified, or set with a value of `true`, suppresses the sending of this parameter to the proxy target-url.
		This property is supported only by the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway then to achieve the same functionality add a **header-control** property to the **invoke** policy configuration in the OpenAPI definition for your API, as in the following examples.
		
		Suppress the **X-Client-ID** header as follows:
		
		`- proxy: target-url: http://myhostname/mypath header-control: type: blacklist values: - ^X-Client-ID$`
		
		Suppress the **userId** query parameter as follows:
		
		`- proxy: target-url: http://myhostname/mypath parameter-control: type: blacklist values: - ^userId$`
			Boolean

x-ibm-gateway-optimize-invoke	No	If set to `false`, prevents the replacement of the last invoke in a policy with proxy. Any value other than `false` (case insensitive) will result in the last invoke in a policy possibly being replaced by proxy when the API is executed in the gateway.	Boolean
x-ibm-gateway-queryparam-encode-plus-char	No	If set to a value of **true**, all "+" characters in the query parameter values of the target-url of Invoke and Proxy policies are encoded to "%2F". The default value is **false**.	Boolean
x-ibm-gateway-api-enforce-response-limits	No	If set to a value of **true**, allows the JSON parser to be enforced on the response rule. If the response body size is higher than the JSON parser limit set in the DataPower domain, a status code of 500 is returned.	Boolean

A list of API properties that control the behavior of the rate limit policy.

Table 4. Properties controlling the rate limit policy

Property	Required	Description	Data type
x-ibm-gateway-emulate-v4-plan-rate-limit	No	By default in IBM API Connect Version 10, if you configure a rate limit only for a Plan and not for the API operations within the Plan then a single rate limit threshold is set for the API as a whole, regardless of which operation in the API is requested. This behavior differs from IBM API Management Version 4.0 where the rate limit is set **individually** for **each** operation in the API. To change the Version 10 behavior to emulate the Version 4.0 behavior, set this API property to a value of **true**.	Boolean

A list of API properties that control the behavior of multiple policies.

Table 5. Properties controlling multiple policies

Property	Required	Description	Data type

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-sourcecode-resolve-apic-variables	No	If set to **true**, API Connect variable references are resolved. Set to **false** if you want the policy to ignore API Connect variable references. The default value is **true**. This property applies to the following policies: • Map • GatewayScript • XSLT • if • switch Note: This property setting is overridden by the x-ibm-gateway-map-resolve-apic-variables API property setting for the Map policy.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-api-json-parse-error-handling	No	If an API request or response payload includes valid JSON content that contains characters that cannot be represented in the JSONX XML internal syntax that is used by the DataPower Gateway, set this property to **escape-unicode** to allow the payload to be accepted without parsing errors. If this property is not configured or is set to any other value, the payload is rejected as invalid JSON. This property applies to the API request payload, and to the API response payload when **x-ibm-gateway-api-enforce-response-limits** is enabled.	String
DataPower Gateway (v5 compatible) only x-ibm-gateway-framework-preserve-escaped-reverse-solidus	No	By default, the string **** in a policy property is converted to a single **** character. Set this property to **true** to preserve the string ****.	Boolean
DataPower Gateway (v5 compatible) only x-ibm-gateway-inspect-request-headers	No	Causes an inspection of the HTTP headers in the API request to check for characters in the header values that are illegal XML characters; it can have the following values: • **default**: There is no inspection for these characters in the header values. If one is present, the API request fails with an HTTP 500 Internal Server Error. • **sanitize**: Any illegal XML characters in header values are replaced with a **?** character. The API processing will continue. Any API that attempts to read **request.headers.<headername>** will see the **?** character in the value. However, the original protocol headers representing **message.headers** will still have the original character, which will be sent to an invoke or proxy backend server. • **bad-request**: There will be an inspection for these characters in the header values. If one is present, the API request fails with an HTTP 400 Bad Request. The default value is **default**.	Boolean

A list of API properties that control the behavior of custom policies.

Table 6. Properties controlling custom policies

Property	Required	Description	Data type
DataPower Gateway (v5 compatible) only x-ibm-gateway-custom-policy-with-gws-action	No	If set to **true**, the **request.body** and **message.body** context variables will be populated for access by an **apim.getvariable('request.body')** or **apim.getvariable('message.body')** function call in a GatewayScript action of a custom policy. If the custom policy does not use a GatewayScript action that requires these variables to be populated, set this property to **false** or do not specify it. The default value is **false**.	Boolean

API policies and logic constructs

Policies and logic constructs are pieces of configuration that control a specific aspect of processing in the Gateway server during the handling of an API invocation at run time.

Policies are the building blocks of assembly flows, and they provide the means to configure capability, such as security, logging, routing of requests to target services, and transformation of data from one format to another. Policies can be configured in the context of an API or in the context of a Plan.

Logic constructs behave in a similar way to policies, but they affect how and which parts of the assembly are implemented without modifying the data flow of the assembly.

IBM® API Connect provides the following ways that you can create, configure, and apply policies and logic constructs:

Policies associated with a Plan
: A Plan provides a mechanism for grouping API operations or subsets of operations from one or more APIs. You can set rate limiting policies on a Plan to specify how many requests an application is allowed to make during a specified time interval. You can also configure a policy for each operation that is included in a Plan. For more information, see Working with Products.

Built-in policies
: A built-in policy enables you to apply a pre-configured policy statement to an assembly to control processing capabilities in the Gateway server. Built-in policies are applied by using the API Designer assembly editor to add a built-in policy to your assembly and to configure the properties for that policy. For more information, see Built-in policies.
Note: You can also apply built-in policies to your APIs by adding an **assembly** extension to your OpenAPI definition file. For more information, see IBM extensions to the OpenAPI specification.

Logic constructs
: A logic construct enables you to control the flow of data through your assembly during an API call. Like policies, logic constructs are applied to an API by using the API Designer assembly editor to add a logic construct to your assembly and to configure the behavior of the construct. For more information, see Logic Constructs.
Note: You can also apply logic constructs to your APIs by adding an **assembly** extension to your OpenAPI definition file. For more information, see IBM extensions to the OpenAPI specification.

User-defined policies

A user-defined policy enables you to create your own policies to control extra processing features in the Gateway server, such as security, or routing of requests. User-defined policies are created outside of API Connect and then imported into one or more Catalogs, so they can be applied to an operation in the same way as built-in policies. For more information, see Authoring policies.

- **Built-in policies**
  IBM API Connect includes a number of built-in policies that you can use to apply preconfigured policy statements to an operation to control an aspect of processing in the Gateway server when an API is invoked.
- **Logic Constructs**
  IBM API Connect includes a number of logic constructs that you can use to apply preconfigured logic to an assembly to control the flow of data through your assembly when the API is called.
- **Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway**
  In the code for GatewayScript and XSLT policies in your API assemblies, you can use API context variables to work with the messages that are generated when an API is called.
- **User-defined policies**
  Make a user-defined policy available to a Catalog, and apply that policy to a REST or SOAP API.

Related reference

- execute

Built-in policies

IBM® API Connect includes a number of built-in policies that you can use to apply preconfigured policy statements to an operation to control an aspect of processing in the Gateway server when an API is invoked.

Note: Although some built-in policies can be used with both the DataPower® Gateway (v5 compatible) and the DataPower API Gateway, some policies are restricted to a particular Gateway. The following icons indicate which Gateway each policy can be used with:

- DataPower Gateway Indicates that the policy can be run on the DataPower Gateway (v5 compatible).
- API Gateway Indicates that the policy can be run on the DataPower API Gateway.

For details of the two types of gateway, see API Connect gateway types.

Built-in policies are configured in the context of an API. You can use the API Designer assembly editor to add a built-in policy to an API and to configure the properties for that policy.

You can also add built-in policies to an API by creating an OpenAPI definition file. For more information, see Creating an OpenAPI definition file.

The following table shows the list of built-in policies that are available. The table contains links to configuration information for both the built-in policy definitions, and the OpenAPI policy definitions. The policies are the same, but they are created in different ways.

Table 1. Built-in policies

Built-in policy	OpenAPI policy	Description	DataPower Gateway	API Gateway
Activity Log	activity-log	Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity. Note: The Activity Log policy is not supported in the assembly for an API whose gateway type is DataPower API Gateway. Instead, you configure activity logging in the API design settings. For details, see Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0).	✓	✓ Functionality provided in the API design; see Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0)
Client Security	client-security	Provides a range of options for authenticating client access to your APIs, extending the capabilities of the OpenAPI specification.	✗	✓
GatewayScript	gatewayscript	Use the gatewayscript policy to execute a specified DataPower GatewayScript program.	✓	✓
Generate JWT	jwt-generate	Use the Generate JWT security policy in IBM API Connect to generate a JSON Web Token (JWT).	✓	✓
Validate JWT	jwt-validate	Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.	✓	✓
if	if	Use the if policy to apply a section of the assembly when a condition is fulfilled.	✓	✓ Functionality provided by switch
GraphQL introspect	graphql-introspect	Use the GraphQL introspect policy to introspect a GraphQL schema.	✗	✓
Invoke	invoke	Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable `message.body` or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.	✓	✓
JSON to XML	json-to-xml	Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.	✓	✓
Log	log	Use the Log policy to customize or override the default activity logging configuration for an API.	✗	✓
Map	map	Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.	✓	✓
operation-switch	operation-switch	Use the operation-switch policy to apply a section of the assembly to a specific operation.	✓	✓

Built-in policy	OpenAPI policy	Description	DataPower Gateway	API Gateway
OAuth	oauth	Use the OAuth policy to policy to perform OAuth processing based on defined OAuth provider settings.	✗	✓
Parse	parse	Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.	✗	✓
Proxy	proxy	Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the `message.body` and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.	✓	✓ Functionality provided by Invoke
Rate Limit	ratelimit	Use the Rate Limit policy to apply one or more rate or burst limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls that an application can make to an API in a specified time period.	✗	✓
Redaction Redaction - DataPower API Gateway Redaction - DataPower Gateway (v5 compatible)	redact - DataPower API Gateway redact - DataPower Gateway (v5 compatible)	Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.	✓	✓
Set Variable	set-variable	Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.	✓	✓
switch	switch	Use the switch policy to execute one of a number of sections of the assembly based on which specified condition is fulfilled.	✓	✓
throw	throw	Use the throw policy to throw an error when it is reached during the execution of an assembly flow.	✓	✓
User Security	user-security	Use the user-security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.	✗	✓
Validate Validate - DataPower API Gateway Validate - DataPower Gateway (v5 compatible)	validate - DataPower API Gateway validate - DataPower Gateway (v5 compatible)	Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.	✓	✓
Validate Username Token	validate-usernametoken	Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.	✓	✗
XML to JSON	xml-to-json	Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).	✓	✓
XSLT	xslt	Use the XSLT policy to apply an XSLT transform to the payload of the API definition.	✓	✓

Including elements in your OpenAPI 2.0 API assembly, Including elements in your OpenAPI 3.0 API assembly

- **Activity Log**
  Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.
- **Client Security**
  The Client Security policy can be used to extend client authentication access for your APIs.
- **Extract**
  Use the Extract policy to extract and transform data from fields in the API context.
- **GatewayScript**
  Use the gatewayscript policy to execute a specified DataPower GatewayScript program.
- **Generate JWT**
  Use the Generate JWT security policy in IBM API Connect to generate a JSON Web Token (JWT).
- **Validate JWT**
  Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.
- **GraphQL introspect**
  Use the GraphQL introspect policy to introspect a GraphQL schema.
- **Invoke**
  Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable `message.body` or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.
- **JSON to XML**
  Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.
- **Log**
  Use the Log policy to customize or override the default activity logging configuration for an API.
- **Map**
  Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.
- **OAuth**
  An OAuth policy performs the requested OAuth processing based on the defined OAuth provider settings.
- **Parse**
  Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.

- **Proxy**
  Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the `message.body` and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.
- **Rate Limit**
  Use the Rate Limit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.
- **Redaction - DataPower API Gateway**
  Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.
- **Redaction - DataPower Gateway (v5 compatible)**
  Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.
- **Set Variable**
  Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.
- **User Security**
  Use the User Security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.
- **Validate - DataPower API Gateway**
  Use the Validate policy to validate the payload in an assembly flow against a schema.
- **Validate - DataPower Gateway (v5 compatible)**
  Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.
- **Validate Username Token**
  Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.
- **Websocket Upgrade**
  Use the Websocket Upgrade policy to process API requests and responses through a WebSocket connection.
- **XML to JSON**
  Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).
- **XSLT**
  Use the XSLT policy to apply an XSLT transform to the payload of the API definition.

Activity Log

Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM® API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see activity-log.

Note that if you are using the DataPower API Gateway, you can configure your logging preferences by using the API design editor. For more information, see Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0).

About

An API event record exists for each API execution event in the Gateway server. By default, the content type that is collected and stored in API event records is `activity` for when API execution completes successfully, and `payload` for when API execution completes with an error code. Apply the Activity Log policy to your assembly to change the type of content to log in these API event records. For more information about API event records, see API event record field reference.
Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled. Enable buffering as follows in the API YAML:

```
x-ibm-configuration:
  ...
  activity-log:
    enabled: true
    error-content: payload
    success-content: payload
  ...
  buffering: true
```

Note:
Activity log policies that call for logging of analytics data upon success do not apply for the OAuth provider. The OAuth provider logs analytics data for failure cases, but does not log successful cases.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Activity Log policy properties

Property label	Required	Description	Data type
Title	Yes	A title for the policy is required, but a default value, `activity-log` is provided.	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
Content	Yes	Defines the type of content to be logged when the operation is successful. Valid values: • **`none`**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **`activity`**: Logs invocation only (only the resource URI is recorded). • **`header`**: Logs activity and header. • **`payload`**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **`activity`**.	string
Error content	No	Indicates what content to log if an error occurs. Valid values: • **`none`**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **`activity`**: Logs invocation only (only the resource URI is recorded). • **`header`**: Logs activity and header. • **`payload`**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **`payload`**.	string

Related tasks

- Creating a new REST OpenAPI definition

Related information

- API Analytics

Client Security

The Client Security policy can be used to extend client authentication access for your APIs.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see client-security.

About

You use the Client Security policy to create an assembly action that's separate from the OpenAPI specification, to allow for more options to authenticate an application. Note that you cannot use the client-security policy to perform rate-limiting enforcement.

In a client security action, you define the following settings.

- Control whether to stop processing if client security fails. When client security fails, stops assembly processing and returns an error.
- Control whether to require the client secret. When required, the secret is compared to the registered secret on the application that is identified by the client ID.
- Define the method to extract client credentials from the request.
  - For all methods except HTTP, use the ID Name and the Secret Name properties to specify the location that contains the ID and the location that contains the secret.
    - When Cookie, specify which cookie.
    - When Context Variable, specify which runtime context variable.
    - When Form, specify the form data.
    - When Header, specify which header.
    - When Query, specify which query parameter.
  - For the HTTP method, use the HTTP Type property to specify the format of the authorization header, which expects the Basic form in the **`basic base64_id:secret`** format.
- Define the method to authenticate the extracted client credentials. The supported methods are Native (this means by using IBM® API Connect), or by using a specified Third party user registry.
  - If Third party, use the User Registry Name property to specify the user-registry to authenticate the extracted client credentials. The supported registry types are LDAP and authentication URL.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Client Security policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`client-security`**.	string

Property label	Required	Description	Data type
Description	No	A description of the policy.	string
Stop on Error	Yes	If this setting is enabled, assembly processing stops if client security fails, and an error is returned. The check box is selected by default.	boolean
Secret Required	Yes	If this setting is enabled, the client secret must be sent in the request. The secret is compared to the registered secret on the application that is identified by the client ID. The check box is selected by default.	boolean
Credential Extraction Method	Yes	Select one of the following options to define how the calling application authenticates: • Header: client ID and client secret credentials must be supplied in the request header. • Query: client ID and client secret credentials must be supplied as query parameters in the request URL. • Form: client ID and client secret credentials must be supplied as form data sent in a POST request. • Cookie: client ID and client secret credentials must be supplied in a header called `Cookie`. • HTTP: the calling application must authenticate by using basic authentication. • Context Variable: the credentials that are used for authenticating the client are obtained from context variables that you set in the assembly flow prior to the Client Security policy, by using a GatewayScript policy for example. The names of these context variables are determined by the values that you supply in the ID Name and Secret Name properties of the Client Security policy. The default value is Header.	string
ID Name	Yes, unless the selected value of Credential Extraction Method is HTTP	The name of the parameter whose value specifies the client ID. For all Credential Extraction Method options other than Context Variable and HTTP, the calling application must supply a parameter with this name, in the location defined by the Credential Extraction Method option. For the Context Variable option, this property specifies the name of a context variable. This property does not apply if the Credential Extraction Method is HTTP.	string
Secret Name	Yes, if Secret Required is enabled and the selected value of Credential Extraction Method is other than HTTP	The name of the parameter whose value specifies the client secret. For all Credential Extraction Method options other than Context Variable and HTTP, the calling application must supply a parameter with this name, in the location defined by the Credential Extraction Method option. For the Context Variable option, this property specifies the name of a context variable. This property does not apply if the Credential Extraction Method is HTTP.	string
HTTP Type	Yes, if the selected value of Credential Extraction Method is HTTP	The authentication type. Currently, the only available option is Basic.	
Authenticate Client Method	Yes	Select one of the following options: • Native: only client ID and client secret are used to authenticate the client. If the selected value of Credential Extraction Method is HTTP then the calling application must supply the client ID for the user name, and the client secret for the password. • Third party: a user registry is used to authenticate the client. If the selected value of Credential Extraction Method is other than HTTP then the calling application must supply the user name for the client ID, and the password for the client secret. The default value is Native.	string
User Registry Name	Yes, if the selected value of Authenticate Client Method is Third party	Select the user registry that will be used to authenticate the client. The supported registry types are LDAP and authentication URL.	string

API Gateway only

Extract

Use the Extract policy to extract and transform data from fields in the API context.

Before you begin

Table 1. Gateways that support this policy and the corresponding policy versions

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly editor. For details about how to configure the policy in your OpenAPI source, see extract.

About this task

The Extract policy specifies the data source that contains the content to transform, the fields that contain the content, and expressions that define how to transform the content. You use a subset of JSONata notation to specify the fields to extract and transform. For more information, see Constructing JSONata expressions to extract and transform data.

The input to the Extract policy must be parsed data. One way to produce parsed data is to add a Parse policy before the Extract policy in your assembly.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Extract policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 2. Extract policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **extract**.	string
Description	No	A description of the policy.	string
Root	Yes	The data source that contains the content to transform. The default value is **message.body**.	string
Capture	Yes	The path expression that identifies the field. The default setting is **$**, which indicates the entire input.	string
Transform	Yes	The expression that defines how to transform the content.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **version** section of the policy YAML. For example:

```
execute:
  - extract:
      version: 2.0.0
      title: extract
...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

Example

Transform the contents of the **account** field to include only the last 4 characters.

```
- extract:
    version: 2.0.0
    title: extract
    root: message.body
    extracts:
      - capture: $.members.policy.**.account
        transform: $substring($,-4)
    description: Include only the last 4 characters of the account field.
```

- **Constructing JSONata expressions to extract and transform data**
  To extract and transform data from fields in the API context when using the Extract policy with the DataPower API Gateway, you supply a JSONata expression that defines the fields to extract and transform.

API Gateway only

Constructing JSONata expressions to extract and transform data

To extract and transform data from fields in the API context when using the Extract policy with the DataPower® API Gateway, you supply a JSONata expression that defines the fields to extract and transform.

Note: For a list of known limitations to DataPower API Gateway support for JSONata, see Limitations to support for JSONata.
The following JSONata functions are supported:

- Aggregation functions:
  - **$average(array)**
  - **$max(array)**
  - **$min(array)**
  - **$sum(array)**
- Array functions:
  - **$append(array1, array2)**
  - **$count(array)**
  - **$reverse(array)**
  - **$sort(array [, function])**
  - **$zip(array1, ...)**
- Boolean functions:
  - **$boolean(arg)**
  - **$exists(arg)**
  - **$not(arg)**
- Numeric functions:
  - **$abs(number)**
  - **$ceil(number)**
  - **$floor(number)**
  - **$formatBase(number [, radix])**
  - **$number(arg)**
  - **$power(base, exponent)**

    - `$round(number [, precision])`
    - `$sqrt(number)`
- Object functions:
    - `$keys(object)`
    - `$lookup(object, key)`
    - `$merge(array<object>)`
    - `$spread(object)`
    - `$type(value)`
      In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: `binary`, `empty`, `graphql`, `stream`, or `xml`.

- String functions:
    - `$contains(str, pattern)`
    - `$join(array[, separator])`
    - `$length(str)`
    - `$lowercase(str)`
    - `$match(str, pattern [, limit])`
    - `$pad(str, width [, char])`
    - `$replace(str, pattern, replacement [, limit])`
    - `$split(str, separator [, limit])`
    - `$string(arg)`
    - `$substring(str, start[, length])`
    - `$substringAfter(str, chars)`
    - `$substringBefore(str, chars)`
    - `$trim(str)`
    - `$uppercase(str)`

- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description

`$apiCtx()`	Generic access to an API context	The `$apiCtx()` extension gives generic access to an API context.
		- Sample transform field in an extract action:
		`"$apiCtx().request.uri"`
		- Sample condition of a case in a switch action:
		`"$apiCtx().request.path = '/simple/apictx-function'"`

`$header(name)`	`message.headers.name`	Message header
`$httpVerb()`	`request.verb`	HTTP method of the request
`$operationID()`	`api.operation.id`	ID of the operation
`$operationPath()`	`api.operation.path`	Path of the operation
`$queryParameter('name')`	- `request.parameters.`*`name`*`.locations` The supported keyword is `query`. - `request.parameters.`*`name`*`.values`	Searches for the index of `query` in `request.parameters.`*`name`*`.locations` and returns `request.parameters.`*`name`*`.values[index]`, where `[index]` is the value for `query` in locations. Parameter values are not URL decoded.
`$statusCode()`	`message.status.code`	Status code
`$storageType([arg])`	*`variable`*`.body` You can specify any variable in the API context. When no variable is specified, the default variable `message.body` is used.	Storage type of the message. The supported values are `binary`, `empty`, `graphql`, `json`, `stream`, or `xml`.

`$urlParameter('name')`	- `request.parameters.`*`name`*`.locations` The supported keywords are `path` and `query`
	- `request.parameters.`*`name`*`.values`
		Searches for the index of `path` and `query` in `request.parameters.`*`name`*`.locations` and returns a single array that contains both `path` and `query` values from `request.parameters.`*`name`*`.values`. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.
		For example, the following URL contains both path and query parameter values.
		`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`
		The `$urlParameter('breed')` URL returns the following array of values.
		`[cats, adopt, Sphynx, Siamese]`
		In this example, the URL includes an API path that is configured as `/petstore/{breed}/{breed}`, where `breed` is configured to be a path parameter of the API path. As a result, `cats` and `adopt` are included in the output.

`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The `xpathExpression` must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. `$xpath($, '//price')`

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description

Extension	Variable	Description
`$gqlActiveOperation([graphql_message])`	`message.body`	Gets the active operation found in the specified GraphQL message. The `operationName` must be the same as the name of the active operation.
`$gqlAlias(graphql_field_node)`	`message.body`	Gets the alias of a GraphQL field node.
`$gqlFragments([graphql_message])`	`message.body`	Gets the fragments found in the specified GraphQL message.
`$gqlName([graphql_node])`	`message.body`	Gets the node name. For operations, the node name is the `operationName`. For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the `operationName` of `message.body` is retrieved.
`$gqlOperations([graphql_message])`	`message.body`	Gets the operations found in the specified GraphQL message.
`$gqlType([graphql_node])`	`message.body`	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- / (division)
- % (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`
  The entire GraphQL query including operations and fragments.

- *`operationName`*
  For an anonymous operation, *`operationName`* can be empty.

- *`~fragmentSpreadName`*
- *`on~typeCondition`*
- *`~~fragmentDefinitionName`*

GatewayScript

Use the gatewayscript policy to execute a specified DataPower® GatewayScript program.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see gatewayscript.

About

The gatewayscript policy gives you built-in access to the DataPower Gateway module via variable **`apim`**.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. gatewayscript policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **gatewayscript**.	string
Description	No	A description of the policy.	string

Source	Yes	The GatewayScript source code to execute. For example:
		
		`var message = [ 'Hello', 'World!' ];`
		`console.debug(message.join(' '));`
			string

Examples

The following examples show how the full OpenAPI for the policy looks in the source code.

Example one:

```
gatewayscript:
  title: writes message to DataPower log
  source: console.debug('Hello World!');
```

Example two:

```
gatewayscript:
  title: script written in multiple lines
  source: |
    var message = [ 'Hello', 'World!' ];
    console.debug(message.join(' '));
```

For more code examples, see GatewayScript code examples and, if you are using the DataPower API Gateway, Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

For general information on using GatewayScript, see the following topics in the DataPower product documentation:

- GatewayScript APIs for API management
- Accessing and manipulating a variable in the API context
- OAuth context variables

Errors

The following error can be thrown while the policy is being executed:

- **JavaScriptError** - a generic error that captures all errors that occur during the execution of the policy.
- **GatewayScript code examples**
  Example GatewayScript code snippets to help the creation of a gatewayscript policy.

Related tasks

- Creating a new REST OpenAPI definition

GatewayScript code examples

Example GatewayScript code snippets to help the creation of a gatewayscript policy.

Attention:

- The mechanisms described here for interacting with the API Connect context are intended for v5-compatible gateways and also to provide v5 compatibility for APIs created for the API Gateway. However, if you are writing a GatewayScript policy or user-defined policy for a new API Gateway API, use the mechanisms described in Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway for better performance and long-term support. For API Gateway APIs, the v5-compatibility functions listed here are to be used for v5 migration only.
- The GatewayScript functions that are listed here use the common name **apim** to call the methods. However, you can change the common name to one of your choice by using one or other of the following function calls depending on your gateway type:

  DataPower Gateway (v5 compatible)
  ```
  var name = require('local:///isp/policy/apim.custom.js');   // v5-Compatible Gateway
  ```
  API Gateway
  ```
  var name = require('apim');                                 // DataPower API Gateway
  ```

  where *name* is your chosen name; for example:

  ```
  var apim = require('apim');
  ```

Access the assembly context

The following code snippets show examples of how to access the assembly context.
Example one returns the **request** context:

```
apim.getvariable('request');
```

Example two returns the header named **myheader**:

```
apim.getvariable('request.headers.myheader');
```

Example three shows how to set, add, or clear a context variable, in this case a message header:

```
apim.setvariable('message.headers.name', value, action)
```

where

- *name* is the name of the message header that you want to set, add, or clear.
- *value* is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the *value* as `request.headers.content-type`. This property is only required when `set` or `add` is specified as the action.
- *action* is the action that you want to apply to the variable. Valid options are:
  - `set`
  - `add`
  - `clear`

  If no option is set, the default option of `set` is applied.

For a complete list of context variables, see API Gateway context variables, and for more information about how to reference context variables in IBM API Connect see Variable references in API Connect. If you are using the DataPower® API Gateway, see also Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

Read input synchronously

The following code snippets show examples of how to read input synchronously by using the `apim.getvariable` function to read data direct from a context. JSON input is returned as a JavaScript object. XML data is returned as a NodeList object (DOM).
Example one returns a synchronous read of the original request body into the variable that is called *json*:

```
var json = apim.getvariable('request.body');
```

Example two returns a synchronous read of the current message into the variable that is called *xml*:

```
var xml = apim.getvariable('message.body');
```

Note: If the content type of the data is neither XML nor JSON, the `apim.getvariable()` function returns a NodeList object consisting of one binary node, which must be converted to a string. If the content really is XML or JSON, you must then parse this string, as illustrated in the following example:

```
var data = apim.getvariable('request.body');

// if a nodelist was returned and the type is 13 (BLOB/Binary Node), convert it
// to a Buffer, which can then be converted to a string
if ((data.item && data.length > 0 && data.item(0).nodeType === 13)) {
  data = data.item(0).toBuffer().toString();
}

// if the string really is XML or JSON, then now add code to parse it with
// the appropriate XML or JSON parse function
      .
      .
      .
```

Read input asynchronously

The following code snippets show the `apim.readInput()` calls that you can use to perform a JavaScript callback function to read input data asynchronously into a variable. JSON input is returned as a JavaScript object. XML data is returned as a NodeList object (DOM). Other types of input are returned as a Buffer object.

```
apim.readInput(callback(err,input) {});
apim.readInputAsJSON(callback(err,json) {});
apim.readInputAsXML(callback(err,nodelist) {});
apim.readInputAsBuffer(callback(err,buffer) {});
```

Note: The `apim.readInput()` function attempts to determine the input data type and call the appropriate `apim.readInputAstype()` function in order to return the data in the correct format. However, to ensure that the correct data type is returned, use the `apim.readInputAstype()` function.
The `apim.readInputAstype()` function reads the data from either the INPUT context (for example `session.INPUT`), or from a policy output context (for example `policy.output`), depending on whether a previous policy had been called that had written output to a policy output context. This function then calls the underlying IBM DataPower GatewayScript `readAstype` method on the relevant context to read the data, and then use JavaScript callbacks to return the data to a variable. For more information about DataPower methods, see Gateway programming model and GatewayScript.

The following is an example of how to use the `apim.readInputAsJSON()` callback function to get data into a variable that is called *json*:

```
apim.readInputAsJSON(function (error, json) {
if (error)
{
// handle error
}
else
{
// the json parameter will contain the json data that has been read
// process the json object in some way and write to the output context
if (json.test=='true')
{
// if json data contains a variable called test that is set to true, then also add some test data
json.data = 'This is my test data'
}
session.output.write(json);
// write to the output context
}
});
```

Configure error information

The following code block shows an example of how to configure the policy implementation to produce error information by using the `apim.error()` function. In this example, `MyError` is thrown to the assembly flow. If there is a catch handler it catches this error, otherwise the assembly skips the execution of the flow and sends a `500 Internal Error` response.

```
apim.error('MyError', 500, 'Internal Error', 'Some error message');
```

where:

- `MyError` is the name of the error.
- `500` is the HTTP code of the required error message.
- `Internal Error` is the HTTP reason phrase for the error.
- `Some error message` is the suggested action for the user.

Accessing the caught exception in a catch block

The following example shows how, in the `catch` block of an API assembly, you can obtain the details of the current caught exception. A possible use would be to create a custom error response using the details of the caught exception.

```
let exception = apim.getError();
```

The function returns a JSON object; for example:

```
{
  "name": "OperationError",
  "message": "This is a thrown Operation Error",
  "policyTitle": "Throw Operation Error",
  "status": {
    "code": "500",
    "reason": "Internal Server Error"
  }
}
```

Write output

The following example shows how to write data into the output, in this case the JSON data `'{ "status": "created" }'`, by using the `session.output.write` variable:

```
session.output.write('{ "status": "created" }');
```

The `session.output.write` variable is a standard GatewayScript method that writes data to the output context. For more information, see Contexts and sessions.
Use the following function to set the format of the content that is written to the `session.output` variable:

```
apim.output('application/type');
```

For example, if you were creating a policy that called:

```
session.output.write('<test>this is some xml data</test>');
```

the policy would write XML data to the output context. You would need to configure the policy to also call

```
apim.output('application/xml');
```

to tell the system that the output is XML. It can cause issues in the processing of the policy if the actual output context format, and the format that is specified in `apim.output`, are different.
Note: If you use `apim.setvariable` to manipulate `message.body`, you must immediately follow the `apim.setvariable` function call with an `apim.output` function call that specifies the content type of the payload that was written to `message.body` by `apim.setvariable`.

Related information

- GatewayScript APIs for API management
- Accessing and manipulating a variable in the API context
- OAuth context variables

Generate JWT

Use the Generate JWT security policy in IBM® API Connect to generate a JSON Web Token (JWT).

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see jwt-generate.

About

JSON Web Token (JWT) is a compact, URL-safe way of representing claims that are to be transferred between two parties. The Generate JWT policy enables you to generate claims and configure whether they are to be used as the payload of a JSON Web Signature (JWS) structure, or as the plain text of a JSON Web Encryption (JWE) structure. Specifying the cryptographic material for both the JWS and the JWE produces a nested JWT that is both digitally signed and encrypted. The JWT is then assigned to the Authorization header as a Bearer token (the default option), or to the runtime variable in the JSON Web Token (JWT) property, if specified.

Note:

- For algorithm types HS256, HS384, and HS512 the cryptographic objects referenced must be a Shared Secret Key.
- For algorithm types RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512 the cryptographic objects referenced must be a Crypto Key (private key).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to sign the JWT.

Prerequisites

The following prerequisites apply:

- If you are using one or more cryptographic objects, they must be located in the API Connect domain on the DataPower appliance. The cryptographic objects must reference the Shared Secret Key or certificate that is needed to encrypt or sign the JWT contents.
- If a JSON Web Key (JWK) is being used, it must be referenced by a runtime variable.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Generate JWT policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`jwt-generate`**.	string
Description	No	A description of the policy.	string
JSON Web Token (JWT)	No	Runtime variable in which to place the JWT that is generated. The default value is: **`generated.jwt`**. However, if not set, the JWT that is generated is written to the Authorization Header as a Bearer token.	string
JWT ID Claim	No	Indicates whether a JWT ID (jti) claim should be added to the JWT. If selected, the property is set to **`true`**, and a UUID is generated and set as the JTI claim value.	boolean
Issuer Claim	Yes	Runtime variable from which the Issuer (iss) claim string can be retrieved. This claim represents the Principal that issued the JWT. The default value is: **`iss.claim`**	string
Subject Claim	No	Runtime variable from which the Subject (sub) claim string can be retrieved.	string
Audience Claim	No	Runtime variable from which the Audience (aud) claim string can be retrieved. Multiple variables are set by using a comma-separated string.	string
Validity Period	Yes	The length of time (in seconds), that is added to the current date and time, in which the JWT is considered valid. The default value is **`3600`**.	integer
Private Claims	No	Runtime variable from which a valid set of JSON claims can be retrieved. These claims are added to any set of claims specified previously.	string
Sign JWK variable name	No	Runtime variable that contains the JWK that is used to sign the JWT.[1]	string
Cryptographic Algorithm	No	The cryptographic algorithm to use. Valid values are: • HS256 • HS384 • HS512 • RS256 • RS384 • RS512 • ES256 • ES384 • ES512 • PS256 • PS384 • PS512	string
Sign Crypto Object	No	The cryptographic object to use to sign the JWT.[1]	string
Encryption Algorithm	No	The encryption algorithm to use. Valid values are: • A128CBC-HS256 • A192CBC-HS384 • A256CBC-HS512	string
Encrypt JWK variable name	No	Runtime variable that contains the JWK to use to encrypt the JWT.	string

Property label	Required	Description	Data type
Key Encryption Algorithm	No	The key encryption algorithm to use. Valid values are: • RSA1_5 • RSA-OAEP • RSA-OAEP-256 • dir • A128KW • A192KW • A256KW	string
Encrypt Crypto Object	No	The cryptographic object to use to encrypt the claim.	string

Example

```
- jwt-generate:
    version: 1.0.0
    title: jwt-generate
    iss-claim: iss.claim
    exp-claim: 3600
    jwt: generated.jwt
    jti-claim: true
    sub-claim: sub.claim
    aud-claim: aud.claim
    private-claims: private.claims
    jws-jwk: jws.jwk
    jws-alg: HS256
    jws-crypto: jwsCryptoObjectName
    jwe-enc: A128CBC-HS256
    jwe-jwk: jwe.jwk
    jwe-alg: A128KW
    jwe-crypto: jweCryptoObjectName
```

Errors

The following error can be thrown while the policy is being executed:

- **`RuntimeError`** - a generic error that captures all errors that occur during the execution of the policy. Upon failure, the detailed error message that is received from the underlying JOSE module is written to the default system log as an error message. This detailed error message is also assigned to the runtime variable jwt-generate.error-message, so it can be retrieved via catch.

If a catch is not configured, in the case of a failure the Generate JWT policy returns an HTTP code 500 **`Invalid-JWT-Generate`** failure. The detailed error message from the underlying JOSE module can be found in the system log.
Attention: If you are an API developer who is troubleshooting a failure that one of your customers had with your API, consider the security risks before sending a customer the exact content of the log message. You can avoid the possibility of someone launching an attack based on the information that they receive from the log message by sending the customer only general information about the message.

Related information

- Introduction to JSON Web Tokens

[1] A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.

Validate JWT

Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see jwt-validate.

About

JSON Web Token (JWT) is a compact, URL-safe way of representing claims that are to be transferred between two parties. The Validate JWT policy enables you to secure access to your APIs by using JWT validation. For example, when an input request that contains a JWT in the header is received, the Validate JWT policy extracts the token, verifies, and decrypts (if appropriate) the signature, and validates the claim. If valid, the claim is put in a runtime variable (for subsequent use if required), and access is allowed to the API. If the claim is not valid, access is denied.

All claims that are specified in the Validate JWT policy are validated, but this is not necessarily all the claims that are contained in the JWT. Not all claims that are in the JWT must be validated, but if any one of the claims that are specified in the Validate JWT policy fail, the whole validation fails. If the validation succeeds, the full set of claims that are contained in the JWT are written to the runtime variable specified in the Output Claims property. Thereby allowing any subsequent action to use this runtime variable to further validate the full set of claims that were in the JWT, as necessary.

Note:

- If the original message was signed with a Shared Secret Key, the cryptographic object that is specified must also be a Shared Secret Key.
- If the original message was signed with a Private Key, the cryptographic object that is specified must be a Crypto Certificate (public certificate).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If a JWK header parameter is included in the header of the JWT, the parameter must match the JWK or cryptographic object that is specified in the policy, or the JWT validation will fail.
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to decrypt or verify the JWT.
- The JWT validate action on the DataPower API Gateway can verify a JWT by using either a single JWK, or a JWK set.

Prerequisites

The following prerequisites apply:

- IBM® DataPower V7.5 with the Application Optimization (AO) option.
- If you are using one or more cryptographic objects, they must be located in the IBM API Connect domain on the DataPower appliance. The cryptographic objects must reference the Shared Secret Key or public certificate that is needed to decrypt the JWT contents or verify the signature.
- If a JSON Web Key (JWK), or JWK set, is being used, it must be referenced by a runtime variable.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate JWT policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`jwt-validate`**.	string
Description	No	A description of the policy.	string
JSON Web Token (JWT)	Yes	Context or runtime variable that contains the JWT to be validated. The default value is: **`request.headers.authorization`**. However, if this property is not set, the policy looks for the JWT in the **`request.headers.authorization`** location by default. Note: The format of the authorization header must be: **`"Authorization: Bearer jwt-token"`** where *jwt-token* is the encoded JWT.	string
Output Claims	Yes	Runtime variable to which the full set of claims that are contained in the JWT is assigned. The default value is: **`decoded.claims`**.	string
Issuer Claim	No	The Perl Compatible Regular Expressions (PCRE) to use to validate the Issuer (iss) claim.	string
Audience Claim	No	The PCRE to use to validate the Audience (aud) claim.	string
Decrypt Crypto Object	No	The cryptographic object (a shared key or certificate) to use to decode the claim.[1]	string
Decrypt Crypto JWK variable name	No	Runtime variable that contains the JWK to use to decrypt the JWT.[1]	string
Verify Crypto Object	No	The cryptographic object (a shared key or certificate) to use to verify the signature.[2]	string
Verify Crypto JWK variable name	No	Runtime variable that contains the JWK, or JWK set, to use to verify the signature.[2]	string

Example

```
- jwt-validate:
    version: 1.0.0
    title: jwt-validate
    jwt: request.headers.authorization
    output-claims: decoded.claims
    iss-claim: "'^data.*'"
    aud-claim: "'^id.*'"
    jwe-crypto: jweCryptoObjectName
    jwe-jwk: jwe.jwk
    jws-crypto: jwsCryptoObjectName
    jws-jwk: jws.jwk
```

Errors

The following error can be thrown while the policy is being executed:

- **`RuntimeError`** - a generic error that captures all errors that occur during the execution of the policy. Upon a validation failure, the detailed error message that is received from the underlying JOSE module is written to the default system log as an error message. This detailed error message is also assigned to the runtime variable jwt-validate.error-message, so it can be retrieved via catch.

If a catch is not configured, in the case of a validation failure the Validate JWT policy returns an HTTP code 500 **`Invalid-JWT-Validate`** failure. The detailed error message from the underlying JOSE module can be found in the system log.

Attention: If you are an API developer who is troubleshooting a failure that one of your customers had with your API, consider the security risks before sending a customer the exact content of the log message. You can avoid the possibility of someone launching an attack based on the information that they receive from the log message by sending the customer only general information about the message.
For examples, see jwt-validate.

Related information

- Introduction to JSON Web Tokens

[1] A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.
[2] A JWK, or JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.

GraphQL introspect

Use the GraphQL introspect policy to introspect a GraphQL schema.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see graphql-introspect.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. GraphQL introspect policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`graphql-introspect`**.	string
Description	No	A description of the policy.	string
Input	No	A variable in the API context that contains the input to introspect. The content of the **`body`** field of the variable is the input to introspect. By default, the variable name is **`message`**.	string
Output	No	A variable in the API context where the results of the introspection are stored. The content of the **`body`** field of the variable is the result of GraphQL introspection. The default variable name is the same as that of the variable specified in the Input field. Therefore, by default, the input of the introspection is overwritten by the output.	string

Invoke

Apply the Invoke policy to call another service from within your assembly. The response from the backend is stored either in the variable **`message.body`** or in the response object variable if it is defined. The policy can be used with JSON or XML data, and can be applied multiple times within your assembly.

Note: The Invoke policy does not support responses with multipart form data, that is, when the response is set to `Content-Type: multipart/related`.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.1 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.3.0 or later)

- **Configuring the Invoke policy for DataPower API Gateway**
  Follow these steps to configure the Invoke policy for DataPower API Gateway in the assembly user interface.
- **Configuring the Invoke policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the Invoke policy for DataPower Gateway (v5 compatible) in the assembly user interface.

Related tasks

- Creating a new REST OpenAPI definition
- Handling errors in the assembly

Related information

- TLS profiles

API Gateway only

Configuring the Invoke policy for DataPower API Gateway

Follow these steps to configure the Invoke policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the Invoke policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the Invoke policy for DataPower Gateway (v5 compatible). For information about the different types of gateway, see API Connect gateway types.
For details on how to configure the policy in your OpenAPI source, see invoke.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Invoke policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Invoke policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`invoke`**.	string
Description	No	A description of the policy.	string
URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Invoke URL value is pre-supplied from information that is defined in the imported WSDL.	string
Backend type	No	Select one of the following options to determine how the payload is sent to the backend. Detect Detect the message type and send the payload to the backend accordingly. XML Package the payload as XML and send. If the message type is not XML, the operation fails. JSON Package the payload as JSON and send. If the message type is not JSON, the operation fails. Binary Package the payload as binary and send, regardless of the message type. GraphQL Package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails. The default value is Detect.	string
GraphQL send type(policy version 2.2.0 and later)	Yes, if Backend type is set to GraphQL or Detect, and HTTP Method is set to POST or Keep	Select one of the following values to determine how the GraphQL payload is sent to the back end. Detect Detect the message type and send the payload to the back end accordingly. GraphQL Package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails. JSON Package the payload as JSON and send. If the message type is not GraphQL, the operation fails. Note: GraphQL send type is supported only if Backend type is set to GraphQL or Detect, and HTTP Method is set to POST or Keep.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply from the endpoint (in seconds). The default value is **`60`**.	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code **`301 Moved Permanently`**. If you select this checkbox, the **`invoke`** policy follows the URL redirection by making a further call to the URL specified in the **`Location`** header in the response. If you clear this checkbox, the **`invoke`** saves the **`301`** status code and the API call is considered to be complete. Note: The **`follow-redirect`** property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the **`invoke`** always follows the URL redirection; the **`proxy`** policy (not supported by the DataPower API Gateway) saves the **`301`** status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string

Property label	Required	Description	Data type
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the Invoke. The following values are valid. • Keep • GET • POST • PUT • DELETE • PATCH • HEAD • OPTIONS The default value is **GET**. However, if set to **Keep**, or the property is removed from the source, the HTTP method from the incoming request is used.	string
HTTP Version (policy version 2.1.0 and later)	Yes	The HTTP version that will be used when connecting to the backend server. Select one of the following values. • HTTP/1.0 • HTTP/1.1 • HTTP/2 The default value is HTTP/1.1.	string
HTTP/2 Required (policy version 2.1.0 and later)	No	Select this checkbox to require that the server selects HTTP/2 during a TLS connection, otherwise the DataPower API Gateway rejects the connection and fails the request. For a non-TLS connection, a warning is logged stating that the requirement can't be enforced. This setting applies only if HTTP Version is set to HTTP/2.	boolean
Compression	No	Select this checkbox to enable Content-Encoding compression on upload. The checkbox is cleared by default.	boolean
Cache Type	No	The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are: Protocol The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL. Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option. No Cache Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings. Time to Live This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached. The default value is Protocol.	string
Time to Live	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to **Time to Live**. Enter a value in the range 5 - 31708800. The default value is **900**.	integer
Cache key	No	Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.	string
Allow WebSocket Upgrade (policy version 2.1.0 and later)	No	Select this checkbox to allow an HTTP or HTTPS connection to be upgraded to the WebSocket protocol if the request received by the corresponding HTTP or HTTPS handler on the DataPower API Gateway uses WebSocket (ws) or WebSocket Secure (wss). Note: • This setting is available only for GraphQL APIs. • The Allow WebSocket Upgrade setting is deprecated. To allow WebSocket upgrade requests that can manage API processing data, define an assembly WebSocket upgrade policy. For more information, see Websocket Upgrade.	boolean
Inject proxy headers	No	If you select this checkbox, the **invoke** policy injects the **X-Forwarded-For**, **X-Forwarded-To**, **X-Forwarded-Host**, and **X-Forwarded-Proto** headers to the request that is sent to the target URL. The checkbox is cleared by default.	boolean
Decode Request Params	No	If you select this checkbox, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The checkbox is cleared by default.	boolean

Property label	Required	Description	Data type
Queryparam Encode	No	If you select this checkbox, all "+" characters in the query parameter values of the target URL are encoded to %2F. The checkbox is cleared by default.	boolean
Keep Payload	No	If you select this checkbox, the invoke policy sends a payload on an **`HTTP DELETE`** method. The checkbox is cleared by default.	boolean
Restrict to HTTP 1.0 (policy version 2.0.0 only)	No	If you select this checkbox, HTTP transactions are restricted to version 1.0. The check box is cleared by default.	boolean
Allow chunked uploads	No	If you select this checkbox, chunked-encoded documents are sent to the server. When the HTTP 1.1 protocol is used, the body of the document can be delimited by either **`Content-Length`** or chunked encoding. While all servers can interpret **`Content-Length`**, many applications fail to understand chunked-encoded documents. For this reason, **`Content-Length`** is the standard method. The use of **`Content-Length`** interferes with the ability of the DataPower Gateway to fully stream. If you must stream full documents to the target server, enable this property. When enabled, the server must be RFC 2616 compatible. Unlike all other HTTP 1.1 features that can be negotiated down at run time, you must know beforehand that the target server is RFC 2616 compatible. The check box is selected by default. Note: Chunked encoding is not supported by the HTTP 1.0 protocol.	boolean
Persistent Connection (policy version 2.3.0 and later)	No	Select this checkbox to enable HTTP persistent connections, allowing connection re-use. The checkbox is selected by default.	boolean
Header control	No	Specifies the headers in **`message.headers`** that you want to copy to the target URL. To prevent headers from being copied, complete the following steps. a. Select Blocklist. b. Click Add blocklist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. To specify headers that you want to be copied, complete the following steps. a. Select Allowlist. b. Click Add allowlist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. The values that you specify are in regular expression format. For example, to specify the Content-Type header, enter **`^Content-Type$`** By default, Blocklist is selected, with no blocklist entries, meaning that **all** headers are copied.	string
Parameter control	No	Specifies the parameters in the incoming request that you want to be copied to the target URL. To prevent parameters from being copied, complete the following steps. a. Select Blocklist. b. Click Add blocklist. c. In the empty field that is displayed, enter the parameter name. d. To add further parameters, repeat the previous steps. To specify parameters that you want to be copied, complete the following steps. a. Select Allowlist. b. Click Add allowlist. c. In the empty field that is displayed, enter the parameter name. d. To add further parameters, repeat the previous steps. The values that you specify are in regular expression format. For example, if the incoming request is **`http://apigw/org/sandbox/petstore/base?petid=100&display=detailed`** and you specify a white list entry of **`^petid$`**, the target URL at run time will be **`http://myhost/mypath?storeid=3&petid=100`** By default, Allowlist is selected, with no allowlist entries, meaning that **no** parameters are copied.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a **`catch`** flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable *message*. Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as Map. Note: If you want the response to be saved in *message*, leave the Response object variable property blank, do **not** supply the value `message`.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **`version`** section of the policy YAML. For example:

```
execute:
  - invoke:
      version: 2.0.0
```

```
      title: invoke
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

Example

```
- invoke:
    version: 2.0.0
    title: get the account status
    target-url: https://example.com/accounts/{id}?status={status}
    cache-response: time-to-live
    cache-putpost-response: true
    tls-profile: MyTLSProfile:1.0.0
    verb: POST
    timeout: 60
    compression: false
    username: MyUser
    password: MyPassword
    stop-on-error:
      - ConnectionError
      - OperationError
```

DataPower Gateway (v5 compatible) only

Configuring the Invoke policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Invoke policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the Invoke policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the Invoke policy for DataPower API Gateway. For information about the different types of gateway, see API Connect gateway types.
For details on how to configure the policy in your OpenAPI source, see invoke.

You might notice that the last invoke in their policy is replaced by a proxy. The replacement is sometimes done automatically by the IBM® API Connect DataPower Gateway to improve performance. The proxy is functionally equivalent to the invoke, but the API caller might notice the following differences when proxy is used.

- If the HTTP request made by the invoke or proxy gets a redirect (`3xx`) response:
  - invoke returns the response from following the redirect response.
  - proxy does not follow `3xx` responses, and the redirect response is returned.
- The IBM API Connect test tool shows that proxy was used, but invoke appears in the analytics latency records.
- The response from the proxy can contain different whitespace or escaping than a response from invoke. Despite the differences in the response, it is still valid.

If you want to prevent replacement of the last invoke in the assembly with proxy, you can set the API property api.properties.x-ibm-gateway-optimize-invoke to `false`. For more information, see API properties.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Invoke policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Invoke policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`invoke`**.	string
Description	No	A description of the policy.	string
URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Invoke URL value is pre-supplied from information that is defined in the imported WSDL.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply from the endpoint (in seconds). The default value is **`60`**.	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code **`301 Moved Permanently`**. If you select this checkbox, the **`invoke`** policy follows the URL redirection by making a further call to the URL specified in the **`Location`** header in the response. If you clear this checkbox, the **`invoke`** saves the **`301`** status code and the API call is considered to be complete. Note: The **`follow-redirect`** property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the **`invoke`** always follows the URL redirection; the **`proxy`** policy (not supported by the DataPower API Gateway) saves the **`301`** status code and completes the API call without following the URL redirection.	boolean

Property label	Required	Description	Data type
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the Invoke. The following values are valid. • Keep • GET • POST • PUT • DELETE • PATCH • HEAD • OPTIONS The default value is **`GET`**. However, if set to **`Keep`**, or the property is removed from the source, the HTTP method from the incoming request is used.	string
Compression	No	Select this checkbox to enable Content-Encoding compression on upload. The checkbox is cleared by default.	boolean
Cache Type	No	The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are: Protocol The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL. Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option. No Cache Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings. Time to Live This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached. The default value is Protocol.	string
Time to Live	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to **`Time to Live`**. Enter a value in the range 5 - 31708800. The default value is **`900`**.	integer
Cache key	No	Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a **`catch`** flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable *message*. Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as Map. Note: If you want the response to be saved in *message*, leave the Response object variable property blank, do **not** supply the value `message`.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **`version`** section of the policy YAML. For example:

```
execute:
  - invoke:
      version: 1.0.0
      title: invoke
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

Example

```
- invoke:
          version: 1.0.0
          title: get the account status
          timeout: 60
          verb: POST
          cache-response: time-to-live
          cache-ttl: 900
          stop-on-error:
            - ConnectionError
            - OperationError
          tls-profile: MyTLSProfile
          target-url: https://example.com/accounts/{id}?status={status}
          username: MyUser
          password: MyPassword
```

JSON to XML

Use the JSON to XML policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

The JSON to XML policy uses a simple convention, based on BadgerFish, to convert your API context payload from JSON to XML. The policy expects the JSON input to be in the same format as the BadgerFish convention, so the structure can be rebuilt in XML. No additional configuration is required. For more information about the BadgerFish convention, see BadgerFish.
Note: The JSON to XML policy does convert the JSON structure `{ "a" : "hello" }` (which is not BadgerFish convention) into `<a>hello</a>`.
Use the IBM® API Connect API Designer assembly view when you are creating your API definition to add a built-in policy to the flow.

The policy must be attached to the flow at the point at which you require the conversion to be performed. For example, if you need to convert a JSON-formatted request into an XML-formatted request, the policy must be attached to the request flow.

The policy reads input from the **`message.body`**, if that context exists, otherwise from the **`request.body`**, and then writes the output to the **`message.body`**.

Note: If you are using the DataPower API Gateway, the input to the JSON to XML policy must be parsed data. One way to produce parsed data is to use a Parse policy before a JSON to XML policy in your assembly flow, which provides explicit control of the parse action.

- **Configuring the JSON to XML policy for DataPower API Gateway**
  Follow these steps to configure the JSON to XML policy for DataPower API Gateway in the assembly user interface.
- **Configuring the JSON to XML policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the JSON to XML policy for DataPower Gateway (v5 compatible) in the assembly user interface.

API Gateway only

Configuring the JSON to XML policy for DataPower API Gateway

Follow these steps to configure the JSON to XML policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the JSON to XML policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the JSON to XML policy for DataPower Gateway (v5 compatible). For more information about the different types of gateway, see API Connect gateway types.
For information about how to configure the policy in your OpenAPI source, see json-to-xml.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API definition that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the JSON to XML policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`json-to-xml`**.	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
Input	No	The input message to convert. Specify the name of a variable in the API context. ***variableName*.body**, the message payload, represents the JSON input to convert. The default value of the variable is **message** and **message.body** is the default input.	string
Output	No	The output message to store the conversion result. Specify the name of a variable in the API context. ***variableName*.body** represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is **message.body**. Otherwise, when the input message is the variable **my-message-variable**, for example, the default output is **my-message-variable.body**. The variable cannot be any read-only in the API context.	string
Conversion type	No	The conversion type that determines the target format of the output. The following options are available: • None: No conversion of the output takes place. • badgerFish: BadgerFish convention is used to determine the target conversion format of the output.	string
Root XML Element Name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the Always output the root element checkbox is selected. The default value is **json**.	string
Always output the root element	Yes	Select this checkbox if you always want the policy to output the root element, even if it is not required to make the XML document well formed. The default value is **false**.	boolean
Element name for JSON array elements	No	The XML element name to be used for JSON array elements.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **version** section of the policy YAML. For example:

```
execute:
  - json-to-xml:
      version: 2.0.0
      title: json-to-xml
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

Example

For example, the following simple JSON object

```
{ "a": { "$" : "hello" } }
```

becomes

```
<a>hello</a>
```

The following JSON object with an attribute

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

becomes

```
<a type="world">hello</a>
```

DataPower Gateway (v5 compatible) only

Configuring the JSON to XML policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the JSON to XML policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the JSON to XML policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the JSON to XML policy for DataPower API Gateway. For more information about the different types of gateway, see API Connect gateway types.
For details on how to configure the policy in your OpenAPI source, see json-to-xml.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the JSON to XML policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Policy properties

Property label	Required	Description	Data type

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is `json-to-xml`.	string
Description	No	A description of the policy.	string
Root XML Element Name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the Always output the root element checkbox is selected. The default value is `json`.	string
Always output the root element	Yes	Select this checkbox if you always want the policy to output the root element, even if it is not required to make the XML document well formed. The default value is `false`.	boolean
Element name for JSON array elements	No	The XML element name to be used for JSON array elements.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the `version` section of the policy YAML. For example:

```
execute:
  - json-to-xml:
      version: 1.0.0
      title: json-to-xml
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

Example

For example, the following simple JSON object

```
{ "a": { "$" : "hello" } }
```

becomes

```
<a>hello</a>
```

The following JSON object with an attribute

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

becomes

```
<a type="world">hello</a>
```

API Gateway only

Log

Use the Log policy to customize or override the default activity logging configuration for an API.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see log.

About

By default, the logging of activity when an API is called, and the sending of the log data to the analytics server, is determined by the settings in the API definition, as described in Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0). However, you can customize activity logging by adding a Log policy to the API assembly flow.

The Log policy enables you to gather all API analytics data for processing in your assembly, and to control what activity log data is sent to the analytics server. Here are some examples of the ways in which you can use the Log policy to customize API activity logging:

- Gather activity log data, use a Redaction policy to redact sensitive fields in the data, then send to the analytics server.
- Send different levels of analytics data depending on certain conditions; for example, the API operation that was called.
- Configure custom activity logging in a global policy that is applied to every API in a Catalog. For more information on global policies, see Working with global policies.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Log policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **log**.	string
Description	No	A description of the policy.	string

Mode	Yes	Select one of the following options:
		• Gather-only: gather all analytics data and write it to the **log** context variable, which populates the API event record on completion of the API execution. For more information on the fields in the **log** context variable, and the consequent API event record, see API event record field reference.
		• Send-only: perform the following actions:
		  ◦ Read the data from the **log** context variable.
		  ◦ Truncate all message payloads and convert to a textual representation.
		  ◦ Send the data to the analytics server.
		• Gather-and-send: perform a **gather-only** operation, immediately followed by a **send-only** operation.
		If you use the Send-only or Gather-and-send option, data is buffered and sent to the analytics server in batches according to the time interval configured for the Analytics Endpoint on the DataPower API Gateway. For more information, see Configuring an analytics endpoint in the DataPower knowledge center.
		Note: If you are offloading to a third party analytics server, you can redact any aspect of the event data. If you are using API Connect analytics, you can redact only request and response payloads.
			string

API Gateway only Log Level	No	The type of content to log. Enter one of the following values: • `none`: Indicates no logging. • `activity`; Logs the invocation only, which is only the resource URI. • `header`: Logs activity and header. • `payload`:Logs activity, header, and payload. • `default`; Use the log level setting defined in the API definition; for more information, see Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0). This is the default value. • `$(value)`: An inline parameter in the format **$(value)** to retrieve a value from the API context.	

Map

Use the Map policy to apply transformations to your assembly flow and specify relationships between variables.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.5.0.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see map.

About

Be aware that MAP decodes all parameter values to an ASCII character equivalent. If a MAP parameter contains entries such as %nn, the MAP output contains decoded values.

For information about the structure of a Map policy and its behavior, see The Map policy structure.

For information about configuring a Map policy by using the user interface, see Configuring the Map policy in the user interface.

For examples of YAML representations of different Map policy configurations, see Map policy examples.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Map policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The field is visible when editing inputs. The default value is **map**.	string
Description	No	A description of the policy. The field is visible when editing inputs.	string
Inputs	Yes	A list of variables that are inputs of the policy.	array (string)
Outputs	Yes	A list of variables that are outputs of the policy.	array (string)
Value	Yes	A GatewayScript program to be performed by the policy in order to map its inputs to its outputs, or to set the value of outputs.	string

Note: The map policy has other properties that are not displayed in the user interface. For a complete list of properties, see map.

- **The Map policy structure**
  The Map policy uses a structure within its OpenAPI definition to specify the behavior of the policy.
- **Configuring the Map policy in the user interface**
  The assemble view in API Designer provides a visual representation of the relations between the inputs and outputs of your Map policy.
- **Map policy examples**
  Examples of the OpenAPI definitions of Map policies.

Related tasks

- Creating a new REST OpenAPI definition

The Map policy structure

The Map policy uses a structure within its OpenAPI definition to specify the behavior of the policy.

This topic contains the following sections:

- Structure
- Input and output definitions
- Actions
- Script
- Fields
- References to inputs and outputs
- Accessing other contexts

For map policy examples, see Map policy examples.

Note: If you deploy your API to the DataPower® Gateway (v5 compatible) then, with the exception of client ID and client secret, the passing of form input as a parameter into an API is not supported. This restriction does not apply if you deploy your API to the DataPower API Gateway.

Structure

In addition to its title and description, a Map policy has the following four main sections:

`inputs`
A list of variables that form the input of the Map policy. Each input has a context variable in which the input variable is found, the variable's name within the Map policy, the content type of the variable, and the definition of the variable or a schema defining its structure.

`outputs`
A list of variables that form the output of the Map policy. These include a context variable where the output variable is found or should be created, the variable's name at output, and the definition of the variable or a schema defining its structure.

`action`
An array containing details of the actions to be performed in order. Each entry includes either a **`set`** or **`create`** field, which specifies the output variable or variables that are part of the action. An action can also contain a **`from`** field, which specifies the input variable or variables that are part of the action.

Each action also contains either a **`value`** field, when the output is set or created, or a **`foreach`** field and a new **`actions`** section, when further actions are nested within the first to set or create elements in a nested array.

`options`
The following property options are available for you to select:

- Include empty. If the check box is selected (the default option), empty XML elements are included in the output of the map policy. Clear the check box if you do not want empty XML elements to be included in the output of the map policy.
- Namespace inheritance. If the check box is selected (the default option), XML namespaces are inherited from the parent element. Clear the check box if you want the map policy to use explicit namespaces.
- Namespace inlining. If the check box is selected (the default option), XML namespaces will be inserted into the document where they are first used. Clear the check box if you want namespaces to all be defined on the root element.
- API Gateway only Resolve XML input data type. If the check box is selected, XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Clear the check box (the default option) if you want all XML element values to be returned as a string.
- API Gateway only Parse XML output: If the check box is selected, the map policy will write XML output to **`message.body`** as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string.
- API Gateway only Empty XML element handling. This property controls how the map policy handles the output of an empty XML element. The following choices are available:
  - String (the default option): An empty XML element is handled as an empty string.
  - Null: An empty XML element is handled as a null value.
  - None: The data is ignored.
  - API Gateway only String Badgerfish: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.
  - API Gateway only Null Badgerfish: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the Allow null value option is selected.
- API Gateway only Use only first array element. If the check box is selected, if an array is encountered in the traversal of the input, only the first element is used. Clear the check box (the default option) if you want the map policy to use all array elements.
- API Gateway only Resolve API Connect variable references. If the check box is cleared (the default option), the map policy ignores API Connect variable references in the map properties. Select the check box if you want API Connect variable references found in the map properties to be resolved.

- API Gateway only Allow null value. If the check box is selected, an input property value with a null value is mapped to the output document. Clear this check box (the default option) if you want the map policy to ignore null input values.
- API Gateway only Optimize schema definition. If the check box is selected, complex schema types evaluation handles circular type references in an optimized manner. Clear this check box (the default option) to evaluate these schema types in a standard manner.
- API Gateway only Create required child properties for array objects and mapped optional objects. If the check box is selected, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:
  - An array consists of objects that contain one or more required properties.
  - An object which is optional has one or more child properties that are required.

  By default, these required properties are not present in the output. If you select this option, these required properties will be present in the output. If the output schema defines a `default` property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:
  - String: empty string ("")
  - Number: 0
  - Boolean: false
  - Object: empty object
  - Array: empty array

  Example 1

  The input data has the following array of objects:

  ```
  [{"a": "value1"}, {"a": "value2", "b": "value3"}]
  ```

  The output schema defines the output object as having two properties, **a** and **b**, of which **b** is required. The map policy defines the following mappings:

  - `input.array.a` to `output.array.a`
  - `input.array.b` to `output.array.b`

  If the check box is selected, and **b** is either not mapped or has no input data present, then **b** is assigned a default value of an empty string, and the output is as follows:

  ```
  [{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]
  ```

  Example 2

  The output schema defines the following structure:

  ```
  {"a" : {"b" : {"c" : "value1", "d" : "value2"} } }
  ```

  Property **b** is optional but property **d** within **b** is required.

  The map policy defines a mapping to `output.a.b.c`.

  If the check box is selected, and **d** is not mapped, then **d** is assigned a default value of an empty string, and the output is as follows:

  ```
  {"a" : {"b" : {"c" : "value1", "d" : ""} } }
  ```

  If the check box is not selected, these required properties are **not** created in the output with their default values.
- API Gateway only Empty JSON array handling. This property controls how the map policy handles the output of an empty array. The following choices are available:
  - All (The default value): Output all empty arrays, including empty children arrays.
  - Parent: Output only the current property's empty array value. Children map actions of this property are not attempted.
  - None: Prevent any empty output array values from being produced.
- API Gateway only Enable JSON post processing. Select this check box to enable post processing of mapped JSON output. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. The check box is cleared by default, meaning that there is no post processing of mapped JSON output.
- API Gateway only Schema definition circular reference limit. Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition.
  The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5.
- Severity level for input data log messages; This property specifies the severity level for log messages that relate to input data. The following choices are available:
  - error
  - warn
  - info
- API Gateway only Create empty parent object for failed mapping. If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Select this check box to create an empty object for the parent of the target mapping, emulating the behavior of IBM® API Management Version 4.0.

  Example

  The map policy defines a mapping to `output.a.b.c`.
  If input data is present, the output is as follows:

  ```
  {
    "a": {
      "b": {
        "c": "inputvalue"
      }
    }
  }
  ```

  If there is no input data, and the Create empty parent object for failed mapping option is selected, the output is as follows:

```
{
  "a": {
    "b": {
    }
  }
}
```

Properties `a` and `b` are created but the value of `b` is an empty object.
The check box is cleared by default.

Input and output definitions

You define the inputs and outputs of your Map policy in their own sections. Each input or output is an element in the array inputs or outputs and is defined by a name, a schema definition or reference, and a variable within the context from which it should be read or to which it should be written. After they have been defined, inputs and outputs are referenced by the name provided in the definition, not by the name of the variable.

The following example shows the inputs and outputs sections of a Map policy.

```
          inputs:
            input_string:
              schema:
                type: string
              variable: request.parameters.name_in
            input_integer:
              schema:
                type: integer
              variable: request.parameters.age_in
          outputs:
            output:
              schema:
                $ref: '#/definitions/output'
              variable: message.body
```

The schema field specifies the schema that describes the variable and can be a simple type, a reference to a definition, or an inline schema definition.

The variable field describes the variable and the context that should be assigned to the input or output variable during the execution of the map policy.

Actions

The fields included in the **actions** section are used in the following ways:

set
: Use the **set** field when you want to assign the result of the **value** field to the output variable specified in the **set** field, replacing the existing value of the output variable. You can specify only one output variable, although this variable can be an array or object.

create
: Use the **create** field when you want to use the result of the **value** field to create a new entry for the output array specified in the **create** field, appending it to the array. You can specify only one output variable, although this variable can be an array or object.

from
: Specify which variables are used in the action as either a single variable or an array of variables, where a variable can be an array or an object. The **from** field is not included if no inputs are used.

value
: Use GatewayScript to provide a script that produces output variables. When a single input is mapped to a single output, the **value** field can be omitted and the variable in **from** is set or created as the variable in **set** or **create** respectively.

default
: Provide a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see Inline references.

foreach
: Specify a variable if you want to execute the associated **actions** field for each entry of the array. The variable can be from the input or output of the Map policy.

actions
: Use the **actions** field to nest actions within an action. Because another action could achieve the same result if applied only once, it is primarily for use with the **foreach** field.

Script

In a **value** or **default** field, use GatewayScript to write the behavior of the action to which the **value** field belongs.

Include the script in single quotation marks. For example: `'4 + 5'` or `'variable_1.toUpperCase()'`.

For information about GatewayScript, see Gateway programming model and GatewayScript

Fields

from, **set**, and **create**
: Each action must have a single **set** or **create** field that specifies the output variable to which the action is applied. Each action can also have a **from** field containing one or more entries that are used to specify the input variable or variables that are used in the action.

  The **set** and **create** fields are both used to assign values to the output variable.

  - **set** replaces the current value of the output variable or creates the variable if it does not already exist.
  - **create** appends a new array entry to the output variable.

  For **set** and **create**, use ***output_variable_name.variable_name*** to specify which of your defined output variables to use, where *output_variable_name* is as defined in the outputs section of the Map policy and *variable_name* refers to an optional field that belongs to the output variable.

For **from**, use ***input_variable_name.variable_name*** to specify which of your defined output variables to use, where *input_variable_name* is as defined in the outputs section of the Map policy and *variable_name* refers to an optional field that belongs to the output variable.

foreach

Use the **foreach** field to specify an input array for which the following **actions** or **value** field will be executed for each entry of the array.

For example:

```
foreach: input.in
actions:
   actions
```

where *input.in* is an input variable that is an array and *actions* is one or more actions in the same format as the parent section. In this example the instructions specified in *actions* are executed once for each array entry of *input.in*.
Referencing the input variable specified in the **foreach** field references the array entry that the current iteration corresponds to.

If a single variable is used in the **foreach** field instead of an array, the following **actions** or **value** field will be applied to or based upon the single variable once and then the loop will terminate.

References to inputs and outputs

Reference variables from the **from** field either by name or by using a number preceded by a '$' and enclosed in parentheses. The variables are numbered from 1, where 1 is the first variable in the array, or the only variable when the **from** field lists only a single variable. For example:

```
value: '$(1) + $(2)'
```

or

```
value: '$(variable_1) + $(variable_2)'
```

where each *variable* is a variable that is included in the **from** field.
During a **foreach** loop, you can reference **$(0)**. The **$(0)** variable begins a **foreach** loop empty but, after an iteration, becomes equivalent to the output of the iteration and can then be referenced again. In this manner, you can apply an array to a single value. For example:

```
- set: out.total
from: in.input
foreach: in.input
value: '$(0) + $(in.input)'
```

where *out.total* is referenced by **$(0)**. In each iteration, the current value of *out.total* and the current array entry of *in.input* are summed, and the value of *out.total* is set as this summation.
When using a foreach to operate on an array, if the elements of the array do not have named fields, you can use **$(this)** to reference the current level of nesting.

Accessing other contexts

At any point within your Map policy's **value** or **default** fields, you can access the context of the API call using the syntax **$(*context.variable*)**.

Alternatively, you can include the variables from other contexts when you define an input to your map policy and then reference it as you would any other input variable.

For a list of available context variables, see API Connect context variables.

Configuring the Map policy in the user interface

The assemble view in API Designer provides a visual representation of the relations between the inputs and outputs of your Map policy.

Procedure

To configure your Map policy, complete the following steps:

1. Click the Map policy in the canvas of the Assemble view.
   The property sheet opens.
2. Click the Edit inputs icon in the Input column.
3. Add an input variable.
   a. Click + input.
   b. In the Context variable field for an input, provide the location of your input variable in the context of the assembly. For a list of context variables, see API Connect context variables.
   c. In the Name field for an input, provide a name for your input for use within only the Map policy.
      Note:
      You must ensure that the name that you provide does not exactly match the value of the Context variable field, or the result might be unpredictable.
   d. Optional: In the Content type field, specify the type of your input. If None is selected, the content type is treated as JSON.
   e. In the Definition field for an input, provide the type of the variable.
      The type can be one from a standard set of types, a definition that you have created for your API, or you can select Inline schema to provide a schema as one of the following options:
      - YAML
      - JSON
      - Generate from sample JSON
      - Generate from sample XML

      .

If you select Object or Array, you can create a schema through the user interface after you have clicked Done and returned to the main view of the property sheet.

4. Optional: To remove a variable, click the corresponding Remove input icon .
5. After you have added all of your input variables, click Done.
6. Click the Edit outputs icon in the Output column.
7. Add an output variable.
   a. Click + output.
   b. In the Context variable field for an output, provide the location of your output. This location can be a new context or one already established during the assembly. For a list of context variables, see API Connect context variables.
   c. In the Name field for an output, provide a name for your variable to use within the Map policy and when it is included in its context at output.
   Note:
   You must ensure that the name that you provide does not exactly match the value of the Context variable field, or the result might be unpredictable.
   d. Optional: In the Content type field, specify the type of your input. If None is selected, the content type is treated as JSON.
   e. In the Definition field for an output, provide the type of the variable.
   The type can be one from a standard set of types, a definition that you have created for your API, or you can select Inline schema to provide a schema as one of the following options:
   - YAML
   - JSON
   - Generate from sample JSON
   - Generate from sample XML

   If you select Object or Array, you can create a schema through the user interface after you have clicked Done and returned to the main view of the property sheet.
8. Optional: To remove a variable, click the corresponding Remove output icon .
9. After you have added all of your output variables, click Done.
10. Optional: If you selected Object or Array for the type of an input or output, create an inline schema definition through the user interface by completing the following steps:
    a. For an Array, click add item. Provide a type for the item and then click the Add icon .
    b. For an Object, click add property. Provide a name and type for the property and then click the Add icon .

    For objects and arrays created in this manner, you can continue to add items and properties, which can themselves be objects and arrays.
11. To connect an input variable to an output variable, click the circle that is directly before the input variable and then click the circle that is directly before the output variable.
    A green line is drawn, linking the two variables together. You can connect multiple inputs to a single output, and a single input can be connected to multiple outputs.
12. To configure an output, whether it has inputs connected to it or not, click the circle directly before the output variable without first clicking on a circle for an input variable.
    The Configure mapping window opens.
13. Optional: In the Mapped from section of the window, you can view which inputs are mapped to the output you are editing. To remove an input, click the Remove input icon alongside the input.
    If the output is part of an array, further configuration options are available. The array, or levels of array in the case of a multidimensional array, can be created by iterating arrays on the input side of the mapping. For each level of your array, select which array on the input side is to be iterated. In the Value field, you can use `$(this)` to reference elements of an array that are not named within the array.
14. Optional: In the Value field, use GatewayScript to configure how any inputs are transformed to produce the output.
    For more information about valid code, see the Script section of the The Map policy structure topic.
15. Optional: In the Default field, provide a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see Inline references.
16. Optional: To delete all mappings to the output, click Delete.
17. When you have configured your outputs, click OK.
18. Optional: Click the Settings icon in the Map column.
    a. Optional: Provide a Title and Description for your Map policy.
    b. To control the XML output of the map policy, select the following options as required:

    Include empty
    : If the check box is selected (the default option), empty XML elements are included in the output of the map policy. Clear the check box if you do not want empty XML elements to be included in the output of the map policy.

    Namespace inheritance
    : If the check box is selected (the default option), XML namespaces are inherited from the parent element. Clear the check box if you want the map policy to use explicit namespaces.

    Namespace inlining
    : If the check box is selected (the default option), XML namespaces will be inserted into the document where they are first used. Clear the check box if you want namespaces to all be defined on the root element.

    API Gateway only Resolve XML input data type
    : If the check box is selected, XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Clear the check box (the default option) if you want all XML element values to be returned as a string.

    API Gateway only Parse XML output
    : If the check box is selected, the map policy will write XML output to **`message.body`** as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string.

    Note: These options effect the XML output only and have no effect on the JSON data.

    c. API Gateway only From the Empty XML element handling list, select one of the following options to control how the map policy handles the output of an empty XML element:
    - String (the default option): An empty XML element is handled as an empty string.
    - Null: An empty XML element is handled as a null value.
    - None: The data is ignored.

- API Gateway only String Badgerfish: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property.
- API Gateway only Null Badgerfish: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the Allow null value option is selected.

d. API Gateway only Select the following general configuration options as required:

Use only first array element
: If the check box is selected, then if an array is encountered in the traversal of the input, only the first element is used. Clear the check box (the default option) if you want the map policy to use all array elements.

Resolve API Connect variable references
: If the check box is cleared (the default option), the map policy ignores API Connect variable references in the map policies. Select the check box if you want API Connect variable references found in the map properties to be resolved.

Allow null value
: If the check box is selected (the default option), an input property value with a null value is mapped to the output document. Clear this check box (the default option) if you want the map policy to ignore null input values.

Optimize schema definition
: If the check box is selected, complex schema types evaluation handles circular type references in an optimized manner. Clear this check box (the default option) to evaluate these schema types in a standard manner.

API Gateway only Enable JSON post processing
: If the check box is selected, post processing of mapped JSON output is enabled. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. The check box is cleared by default, meaning that there is no post processing of mapped JSON output.

API Gateway only Create empty parent object for failed mapping
: API Gateway only Use this setting to control the behavior if a mapping fails because its input is not present and there is no default mapping configured.
The default behavior is to make no change to the output, but if you select this check box an empty object is created for the parent of the target mapping, emulating the behavior of IBM API Management Version 4.0.

  Example
  : The map policy defines a mapping to `output.a.b.c`.
  If input data is present, the output is as follows:

```
{
  "a": {
    "b": {
      "c": "inputvalue"
    }
  }
}
```

  If there is no input data, and the Create empty parent object for failed mapping option is selected, the output is as follows:

```
{
  "a": {
    "b": {
    }
  }
}
```

  Properties `a` and `b` are created but the value of `b` is an empty object.
  The check box is cleared by default.

Create required child properties for array objects and mapped optional objects
: If the check box is selected, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:
  - An array consists of objects that contain one or more required properties.
  - An object which is optional has one or more child properties that are required.

  By default, these required properties are not present in the output. If you select this option, these required properties will be present in the output. If the output schema defines a `default` property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:
  - String: empty string ("")
  - Number: 0
  - Boolean: false
  - Object: empty object
  - Array: empty array

  Example 1
  : The input data has the following array of objects:

```
[{"a": "value1"}, {"a": "value2", "b": "value3"}]
```

  The output schema defines the output object as having two properties, `a` and `b`, of which `b` is required. The map policy defines the following mappings:

  - `input.array.a` to `output.array.a`
  - `input.array.b` to `output.array.b`

  If the check box is selected, and `b` is either not mapped or has no input data present, then `b` is assigned a default value of an empty string, and the output is as follows:

```
[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]
```

Example 2

The output schema defines the following structure:

```
{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }
```

Property **b** is optional but property **d** within **b** is required.

The map policy defines a mapping to **output.a.b.c**.

If the check box is selected, and **d** is not mapped, then **d** is assigned a default value of an empty string, and the output is as follows:

```
{"a" : {"b" : {"c" : "value1", "d" : ""} } }
```

If the check box is not selected, these required properties are **not** created in the output with their default values.

e. API Gateway only From the Empty JSON array handling list, select one of the following options to control how the map policy handles the output of an empty array:
   - All (the default option): Output all empty arrays, including empty children arrays.
   - Parent: Output only the current property's empty array value. Children map actions of this property are not attempted.
   - None: Prevent any empty output array values from being produced.

f. From the Severity level for input data log messages list, select one of the following options to specify the severity level for log messages that relate to input data:
   - error
   - warn
   - info

g. API Gateway only Set the Schema definition circular reference limit field to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5.

h. Click Save when done.

Results

You have configured a Map policy to transform and map variables in your assembly flow.

Map policy examples

Examples of the OpenAPI definitions of Map policies.

- One to one mapping
- Many to one mapping
- A simple transformation using the value field
- Mapping from multiple contexts into a new context
- Mapping to an inline schema definition
- Mapping with a default value
- Mapping an array into a single value
- Mapping array elements to an array
- Using the advanced XML options

One to one mapping

The following example:

- maps parameters from the request's query to an object in the message body.
- maps one strings directly to a single string.
- maps one integer directly to a single integer.

The referenced definition, **output**, defines an object containing a string, name, and an integer, age.

```
      - map:
          title: 1-1 map
          inputs:
            input_string:
              schema:
                type: string
              variable: request.parameters.name_in     # the location of the variable, named "name_in" and found in the query
parameters of the request
            input_integer:
              schema:
                type: integer
              variable: request.parameters.age_in    # another variable in the query parameters of the request
          outputs:
            output:
              schema:
                $ref: '#/definitions/output'    # a schema definition reference to use for the output. The schema is for the
whole of the message body.
              variable: message.body
          actions:
            - set: output.name_out    # in the actions section, variables are referenced by their name within the map policy
              from: input_string      # if a value field is not used, the mapping is direct with the input value used for the
output variable
            - set: output.age_out     # because the 'output' variable is itself an object, further references are made to
variables that it contains
              from: input_integer
```

Many to one mapping

The following example:

- maps parameters from the request's query to an object in the message body.
- maps two strings to a single string by concatenating them.
- maps two integers to a single integer by summing them.

The referenced definition, `output`, defines an object containing a string and an integer.

```
    - map:
        title: many-1 map
        inputs:
          input_string_1:
            schema:
              type: string
            variable: request.parameters.first_name
          input_string_2:
            schema:
              type: string
            variable: request.parameters.last_name
          input_integer_1:
            schema:
              type: integer
            variable: request.parameters.balance_1
          input_integer_2:
            schema:
              type: integer
            variable: request.parameters.balance_2
        outputs:
          output:
            schema:
              $ref: '#/definitions/output'
            variable: message.body
        actions:
          - set: output.full_name
            from:
              - input_string_1
              - input_string_2
             value: |
               var retValue = undefined;
               if ($(input_string_1) !== undefined && $(input_string_2) !== undefined) {
                 retValue = $(input_string_1).toUpperCase() + ' ' + $(input_string_2).toUpperCase()"
               }
               retValue;
          - set: output.total_balance
            from:
              - input_integer_1
              - input_integer_2
            value:  |
              var i1 = 0;
              var i2 = 0;
              if ($(input_integer_1) !== undefined) i1 = $(input_integer_1);
              if ($(input_integer_2) !== undefined) i2 = $(input_integer_2);
              i1 + i2;
```

A simple transformation using the value field

The following example:

- maps a parameter from the request's query to an object in the message body.
- maps one string featuring lowercase characters to a single string containing only uppercase characters.

The referenced definition, `output`, defines an object containing a single string.

```
    - map:
        title: Uppercase map
        inputs:
          input_lowercase:
            schema:
              type: string
            variable: request.parameters.name_in
        outputs:
          output_uppercase:
            schema:
              $ref: '#/definitions/output'
            variable: message.body
        actions:
          - set: output_uppercase.name_out
            from: input_lowercase
            value: $(input_lowercase).toUpperCase()    # the input variable is referenced and calls the toUpperCase method to
produce a value for the output variable
```

Mapping from multiple contexts into a new context

The following example:

- maps an integer from the request's header to an integer in a new message body.
- maps a string from the message's body to the body of a new custom context, named *new_context*.

```
      - map:
          title: Context map
          inputs:
            input_integer:
              schema:
                type: integer
              variable: request.headers.age_in      # the 'age_in' header of the request is used as an input
            input_string:
              schema:
                type: string
              variable: message.body.name_in      # as is the 'name_in' field of the request body
          outputs:
            output_integer:
              schema:
                type: integer
              variable: message.body.age_out
            output_string:
              schema:
                type: string
              variable: new_context.body.name_internal    # the context named 'new_context' is created by the map policy and
exists only while the assembly is processed
          actions:
            - set: output_string
              from: input_string
            - set: output_integer
              from: input_integer
```

Mapping to an inline schema definition

The following example:

- maps parameters from the request's headers to an object in the message body, which is defined within the map policy.
- maps one string directly to a single string.
- maps one integer directly to a single integer.

```
      - map:
          title: Inline schema map
          inputs:
            input_integer:
              schema:
                type: integer
              variable: request.headers.age_in
            input_string:
              schema:
                type: string
              variable: request.headers.name_in
          outputs:
            output:
              schema:                    # instead of a simple type or a reference to a definition, an inline YAML definition
is used
                type: object
                properties:
                  name_out:
                    type: string
                    name: name_out
                  age_out:
                    type: integer
                    format: int32
                    name: age_out
                title: output
              variable: message.body
          actions:
            - set: output.age_out
              from: input_integer
            - set: output.name_out
              from: input_string
```

Mapping with a default value

The following example:

- maps one string directly to a single string.
- provides a default value for the output string if a valid input string is not provided.

```
      - map:
          title: Default Map
          inputs:
            input_string:
              schema:
                type: string
              variable: request.headers.name_in
          outputs:
            output_string:
              schema:
                type: string
              variable: message.body.name_out
          actions:
            - set: output_string
              from: input_string
              default: John Smith            # the default field is specified in the same way as a value, in this case
providing a fixed value
```

Mapping an array into a single value

The following map policy:

- maps a single array of integers into a single integer.
- sums the integers in the array.
- `$(0)` represents the accumulated output because map evaluates all array element values.

```
      - map:
          title: Summation map
          inputs:
            input:
              schema:
                $ref: '#/definitions/balance_array_in'
              variable: request.body
          outputs:
            output:
              schema:
                type: integer
              variable: message.body.total_balance_out    # instead of using a full schema definition of the message body, a
single variable in the message body is specified
          actions:
            - set: output
              from: input
              foreach: input      # the foreach field specifies that each value of the array 'input' is to be iterated over
              value: $(0)+$(input) # the $(0) reference is the accumulated value of the 'output' variable
```

Mapping array elements to an array

The following map policy:

- maps an array, whose elements are objects containing two integers, to an array, whose entries contain a single integer field.
- maps an array to an array of the same length.
- takes the difference of the values of the integers in each array element to create a single integer in each array element.

```
      - map:
          title: Array summation
          inputs:
            input:
              schema:
                $ref: '#/definitions/balance_and_credit_array'
              variable: request.body
          outputs:
            output_array:
              schema:
                type: array
              variable: message.body
          actions:
            - create: output_array
              from: input
              foreach: input
              actions:
                - set: total_balance_out
                  from:                          # inside the actions section, variables inside the array elements being
iterated over are used
                    - integer_in_1
                    - integer_in_2
                  value: $(integer_in_1)-$(integer_in_2)      # the difference of the variables is taken for each array entry
of 'input'
```

Using the advanced XML options

The following example:

- does not include empty elements in the output of the map policy.
- uses explicit name spaces rather than inheriting name spaces from the parent element
- indicates that all namespace declarations will be placed on the root XML element.
- uses all array elements rather than only the first
- ignores API Connect variable references
- maps input property values with a null value to the output document
- handles circular type references in an optimized manne

```
        actions:
            - set: output, two
              from: input, one
        options:
            includeEmptyXMLElements: false
            namespaceInheritance: false
            inlineNamespaces: false
            mapArrayFirstElementValue: false
            mapResolveApicVariables: false
            mapNullValue: true
            mapOptimizeSchemaDefinition: true
            mapCreateEmptyArray: parent
```

OAuth

An OAuth policy performs the requested OAuth processing based on the defined OAuth provider settings.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see oauth.

About

By adding an assembly OAuth policy, you specify the OAuth provider settings to use to perform the requested OAuth processing and the supported OAuth processing components.

Note: You add an OAuth policy to the assembly in a native OAuth provider. For more information, see the following topics:

- Editing the native OAuth provider configuration using the API Editor (Cloud Manager UI)
- Editing the native OAuth provider configuration using the API Editor (API Manager UI)

You can specify the settings through one or more of the following methods.

- URL reference
- Literal configuration
- Object reference

When you use more than one method to specify the same aspect of the OAuth provider settings, the following precedence rules apply to enforce the settings dynamically for the incoming transactions.

- URL reference takes precedence over any existing literal configuration or object reference.
- Literal configuration takes precedence over any existing object reference.

You can configure the policy to support one or more of the following processing components:

Validate request
: Validates the authorization request from the client.

Generate authorization code
: Generates the authorization code for the client, which represents the resource owner's authorization that grants access to the requested resource.

Verify authorization code
: Verifies the authorization code from the client.

Verify refresh token
: Verifies the refresh token that is presented by the client.

Generate access token
: Generates the access token to the client when the authorization code or refresh token is verified.

Introspect token
: Introspects the token to determine its state and, when active, its metadata.

When the policy does not support a processing component but that processing is requested, the unsupported component is not run.

You can include multiple OAuth policies in your OAuth provider assembly. For example, your assembly might include the following flow:

1. OAuth policy that validates the request.
2. GatewayScript policy.
3. OAuth policy that generates authorization code.

For more information, see Example - using multiple OAuth policies in an OAuth provider assembly and OAuth context variables.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. OAuth policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`oauth`**.	string
Description	No	A description of the policy	string
Default OAuth Provider Settings Object	Yes	The name of an existing OAuth provider that defines the required settings.	string
Dynamic OAuth configuration from a URL	No	A URL to a document that contains serialized XML or JSON properties that defines OAuth token generate settings.	string
Dynamic OAuth configuration from a literal string	No	A literal string that contains serialized XML or JSON properties that defines OAuth token generate settings.	string
Supported OAuth components	No	Select the OAuth components that are supported by this policy.	string

Overriding the default OAuth provider settings

You can use either the Dynamic OAuth configuration from a literal string property or the Dynamic OAuth configuration from a URL property to dynamically override any OAuth provider configuration settings defined by the Default OAuth Provider Settings Object property.

For example, to override the access token expiration time with a value of 200 seconds, include the following configuration in either the literal string or the document at the specified URL:

```
<OAuthProviderSettings><APICAccessTokenTTL>200</APICAccessTokenTTL></OAuthProviderSettings>
```

For a list of all OAuth provider settings, refer to the **OAuthProviderSettings** management schema, defined in the xml-mgmt.xsd file located in the store: directory on the DataPower API Gateway.

If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

- **Example - using multiple OAuth policies in an OAuth provider assembly**
  This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

Related tasks

- Creating a new REST OpenAPI definition

Related reference

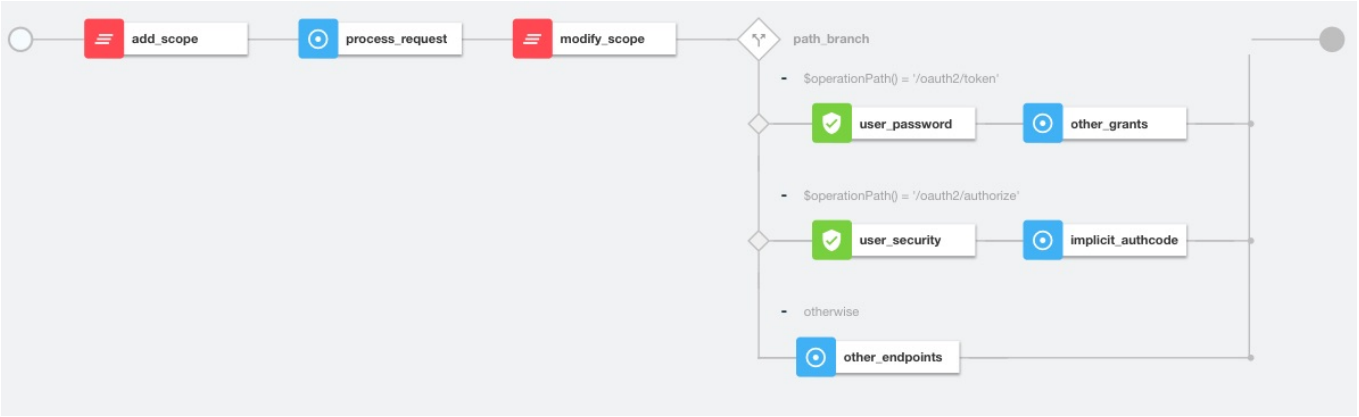
- API Connect context variables

Example - using multiple OAuth policies in an OAuth provider assembly

This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

The example is based on the default assembly that is generated when you create a native OAuth provider, and is customized with the addition of **gatewayscript** policies that use OAuth context variables to manipulate the OAuth flow. For details on creating a native OAuth provider, see Configuring a native OAuth provider.

It has the following assembly flow:



The following sections describe the OpenAPI source code that underlies each of the policies in the assembly; for the complete assembly code, download multiple_oauth_policies.txt.

Sample policy to add a custom scope

The **add_scope** policy is a **gatewayscript** policy that adds a custom scope to the request.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
    version: 2.0.0
    title: add_scope
    source: |-
      // Add another custom scope to the request
      let scope = context.get("request.parameters.scope.values[0]);
      if (scope)
        context.set("oauth.processing.scope", scope + " custom");
```

Validate the initial OAuth request

The first **process_request** policy is an **oauth** policy that processes the initial request and verifies that the request is valid. The result of the processing is stored automatically in the oauth.processing context variables for use, as required, by the next OAuth policy in the assembly flow.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
    title: process_request
```

```
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect protocol steps
      that are needed for OAuth Validation by default. The inputs and
      outputs of each of the steps are driven by documented context
      variables. Add or remove the Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthValidateRequest
```

Sample policy to modify the scope

The **modify_scope** policy is a **gatewayscript** policy that modifies the scope depending on the calling application.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
    version: 2.0.0
    title: modify_scope
    source: |-
      let admin_id = '1f1a2aa4-db9f-4423-b2f1-e2572b12123a';

      // Check application and modify the scope
      let app = context.get("oauth.processing.client_id");
      let scope = context.get("oauth.processing.scope");
      if (app === admin_id) {
        context.set("oauth.processing.scope", scope + " admin");
      } else {
        context.set("oauth.processing.scope", scope + " customer");
      }
```

Branch conditionally according to the OAuth path

The **path_branch** policy is a **switch** policy that branches according to the different OAuth paths to process the resource owner.

The underlying OpenAPI source YAML is as follows:

```
- switch:
    version: 2.0.0
    title: path_branch
    case:
      - condition: ($operationPath() = '/oauth2/token')
        execute:
            .
            .
            .
          definition of the user_security and other_grants policies
            .
            .
            .
      - condition: ($operationPath() = '/oauth2/authorize')
        execute:
            .
            .
            .
          definition of the user_password and implicit_authcode policies
            .
            .
            .
      - otherwise:
            .
            .
            .
          definition of the other_endpoints policy
            .
            .
            .
```

Process the user name and password, and enable grant type component

The following two policies operate on the token endpoint.

- The **user_password** policy for password grant type processes the user name and password from the **x-www-form-urlencoded** body. The authentication method is derived from the User Security settings in the OAuth provider; see Configuring user security for a native OAuth provider.
- The **other_grants** policy is an **oauth** policy that enables the **OAuthGenerateAccessToken**, **OAuthVerifyAZCode**, **OAuthVerifyRefreshToken**, and **OAuthCollectMetadata** components to perform the operations for client credentials, authorization code, refresh token, and password grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
    title: user_password
    version: 2.0.0
    description: ''
    factor-id: default
    extract-identity-method: context-var
    user-context-var: request.parameters.username.values
    pass-context-var: request.parameters.password.values
    ei-stop-on-error: false
    user-auth-method: auth-url
```

```
    au-stop-on-error: false
    auth-url: 'http://httpbin.org/basic-auth/user/pass'
    user-az-method: authenticated
    az-stop-on-error: true
    auth-response-headers-pattern: (?)x-api*
    auth-response-header-credential: X-API-Authenticated-Credential
- oauth:
    title: other_grants
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for token path by default.
      The inputs and outputs of each of the steps are driven by
      documented context variables. Add or remove the Supported
      OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthGenerateAccessToken
      - OAuthVerifyAZCode
      - OAuthVerifyRefreshToken
      - OAuthCollectMetadata
```

Perform authorization checks, and enable grant type components

These following two policies operate on the authorize endpoint.

- The **user_security** policy configuration is derived from the User Security settings in the OAuth provider; see Configuring user security for a native OAuth provider.
- The **implicit_authcode** policy is an **oauth** policy that enables the **OAuthGenerateAZCode**, **OAuthGenerateAccessToken**, and **OAuthCollectMetadata** components to perform the operations for the implicit and authorization code grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
    title: user_security
    version: 2.0.0
    description: >-
      This user security policy performs EI(basic) and AU(auth
      url) check for oauth assembly. Change the security check
      method as required
    factor-id: default
    extract-identity-method: basic
    ei-stop-on-error: true
    user-auth-method: auth-url
    au-stop-on-error: true
    user-az-method: authenticated
    az-stop-on-error: true
    auth-response-headers-pattern: (?)x-api*
    auth-response-header-credential: X-API-Authenticated-Credential
    auth-url: 'http://httpbin.org/basic-auth/user/pass'
- oauth:
    title: implicit_authcode
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for az code path by
      default. The inputs and outputs of each of the steps are
      driven by documented context variables. Add or remove the
      Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthGenerateAZCode
      - OAuthGenerateAccessToken
      - OAuthCollectMetadata
```

Process all other endpoints

The **otherwise** condition catches all other endpoints such as the introspect and revoke endpoints. The **other_endpoints** policy in the **otherwise** condition is an **oauth** policy that enables the **OAuthIntrospectToken**, and **OAuthRevokeTokencomponents** components to perform the operations for introspect and revoke.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
    title: other_endpoints
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for all other paths by
      default. The inputs and outputs of each of the steps are
      driven by documented context variables. Add or remove the
      Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthIntrospectToken
      - OAuthRevokeToken
```

Related concepts

- API policies and logic constructs

Related reference

- OAuth context variables

Related information

- Configuring a native OAuth provider

Parse

Use the Parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later) 2.2.0 (DataPower API Gateway Version 10.5.0.5 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see parse.

About Parse

With the parse policy, you can retrieve parse settings by using the following methods in increasing order of precedence.

1. Specify a valid parse settings configuration from which to retrieve its properties as the default parse settings.
   Note: If you change the default name, you must separately configure a corresponding ParseSettings object on the DataPower API Gateway.
2. Specify a literal string as serialized XML or JSON properties as parse settings. These properties take precedence over any existing default properties. The literal string must be XML or JSON formatted.
   - The XML format is `"<ParseSettings><propertyName>propertyValue</propertyName></ParseSettings>"`.
   - The JSON format is `"{ \"ParseSettings\" : { \"propertyName\" : propertyValue } }"`.
3. A URL that represents a named context from which to retrieve the serialized XML or JSON properties as parse settings. These properties take precedence over any existing literal or default properties.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Parse policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **parse**.	string
Description	No	A description of the policy.	string
Use Content Type	No	If this setting is enabled and the parse setting is configured to detect the document type, the parse operation uses the **Content-Type** specified in the request headers. If this setting is enabled and the document type in the parse setting is configured for either JSON or XML, the parse operation uses the **Content-Type** specified in the request headers and fails if the **Content-Type** in the request headers does not match the parse setting. Enabling this setting is applicable only when the expected **Content-Type** is either JSON or XML. If this setting is not enabled, the parse operation uses either the document type that is specified in the parse setting, or the detected document type if the parse setting is configured to detect the document type. The check box is cleared by default.	boolean
Parse settings object reference	No	An existing valid object from which to retrieve default property values for the dynamic object.	string

Property label	Required	Description	Data type

Document type	No	The type of document to parse. Select one of the following options:
		• Detect: the document type is detected and the payload is parsed accordingly. This type is the default option.
		• JSON: the payload is parsed as JSON.
		• XML: the payload is parsed as XML.
		• Graphql: parsing is performed in one of the following ways:
		◦ As GraphQL in the body of a POST request. The GraphQL is in the following format:
		`{ me { name } }`
		◦ As a JSON string in the body of a POST request that contains a GraphQL query. The JSON object is in the following format:
		`{ "query": "query MyData ($idVar: Int!) { user (id: $idVar) { name }}", "operationName": "MyData", "variables": {"idVar": 123} }`
		◦ Through the URL query of a GET request. The URL is in the following format:
		`https://myapi/graphql?query={me{name}}`
		The following example includes the **variables** and **operationName** properties, with URL encoding (line-breaks are for table formatting only, your value will not include line-breaks):
		`https://hostname/basepath/graphql?query=query%20fetchAccounts %20($limit:%20Int)%20{accounts(limit:%20$limit)%20{name{first,last}}} &variables={"limit":100}&operationName=fetchAccounts`
		Query variables in a GET request can be sent as a JSON string in a query parameter called **variables**. If the query contains several named operations, an **operationName** query parameter is required to determine which operation is to be executed.
			string

Maximum document size	No	Optionally, the maximum document size in bytes of a JSON, XML, or GraphQL input document that the parse operation accepts. This setting provides threat protection by enforcing the maximum document size. Enter a value in the range 0 - 5368709121. A document is rejected when its size exceeds the maximum size. A value of 0 indicates an unlimited document size. When set to 0, the operation result does not return the document size. The default value is 4194304.	integer
Maximum nesting depth	No	Optionally, the maximum nesting depth of an XML, JSON, or GraphQL message that the parse operation accepts. This setting provides threat protection by enforcing limits in the message. • When XML, it is the maximum level of element depth. • When JSON, it is the maximum level of nested label-value pairs, the maximum number of nested arrays, or the maximum number of combination of label-value pairs and arrays. • When GraphQL, it is the maximum level of nested selection sets. Enter a value in the range 0 - 4096. A document is rejected when its nesting depth exceeds the maximum depth. A value of 0 indicates unlimited nesting depth. When set to 0, the operation result does not return the nesting depth. The default value is 512.	integer
Maximum width	No	Optionally, the maximum width of the payload that the parse operation accepts. • When the input document is XML, this property specifies the maximum number of attributes for an element or the maximum number of child elements for an element. • When the input document is JSON, this property specifies the maximum number of properties on a JSON object or the maximum number of JSON items in a JSON array. • When the input document is GraphQL, this property specifies the maximum number of selections in a selection set. Enter a value in the range 0 - 65535. The document is rejected when its width exceeds the maximum width. A value of 0 indicates unlimited width. When set to 0, the operation result does not return the width of the document. The default value is 4096.	integer
Maximum name length	No	Optionally, the maximum name length in bytes in a document that the parse operation accepts. • For XML, it is the length of the name portion of a tag. • For JSON, it is the length of the label portion of the JSON label-value pair. • For GraphQL, it is the maximum length of the identifiers, including field names and directive names. The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 8192. A document is rejected when its name length exceeds the maximum length. A value of 0 indicates unlimited name length. When set to 0, the operation result does not return the name length of a document. The default value is 256.	integer

Property label	Required	Description	Data type
Maximum value length	No	Optionally, the maximum value length in bytes in a document that the parse operation accepts. • For XML, it is the length of an attribute or the length of a text value. • For JSON and GraphQL, it is the length of a string value. The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 5368709121. A document is rejected when its value length exceeds the maximum length. A value of 0 indicates unlimited value length. When set to 0, the operation result does not return the value length of a document. The default value is 8192.	integer
Maximum number of unique names	No	Optionally, the maximum number of unique names in a JSON or XML document that the parse operation accepts. • For XML, it is the number of unique XML local names. • For JSON, it is the number of unique JSON labels. Enter a value in the range 0 - 1048575. A document is rejected when the number of unique names in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique names. When set to 0, the operation result does not return the number of unique names in a document. The default value is 1024.	integer
Maximum number of unique prefixes	No	Optionally, the maximum number of unique XML namespace prefixes in a document that the parse operation accepts. This limit counts multiple prefixes defined for the same namespace, but does not count multiple namespaces defined in different parts of the input document under a single prefix. Enter a value in the range 0 - 262143. A document is rejected when the number of unique prefixes in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique prefixes. When set to 0, the operation result does not return the number of unique prefixes in a document. The default value is 1024.	integer
Maximum number of unique namespaces	No	Optionally, the maximum number of unique XML namespace URIs that the parse operation accepts. This limit counts all XML namespaces, regardless of how many prefixes are used to declare them. Enter a value in the range 0 - 65535. A document is rejected when the number of unique namespaces in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique namespaces. When set to 0, the operation result does not return the number of unique namespaces in a document. The default value is 1024.	integer
Maximum number length	No	Optionally, the maximum number of bytes in the value portion of a JSON label-value pair when the value is a number. The number must be a contiguous string of bytes that contain no white space. The number can include a minus sign and a positive or negative exponent. Enter a value in the range 0 - 256. A document is rejected when the number length in the document exceeds the maximum length. A value of 0 indicates unlimited number length. When set to 0, the operation result does not return the number length in a document. The default value is 128.	integer
Parse settings literal configuration	No	A literal string as serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing default properties.	string
Parse settings URL reference	No	A URL that represents a named context from which to retrieve the serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing literal or default properties.	string
Message for parsing	No	The name of a variable in the API context. The content of the **`body`** field of the variable is the input to the policy. The default variable name is **`message`**.	string
Message for resulting parsed data	No	The name of a variable in the API context. The content of the body field of the variable is the output of the parse operation. The parse metrics of the parsed document can be stored in a different part of the message. The default variable name is the same as the input name, so by default the input message is overwritten by the output message.	string

Use the YAML source to set the following option.

Property	Required	Description	Data type
strict-utf8-encoding	No	For JSON documents, whether to enforce strict UTF-8 encoding throughout the entire document. • When **`on`**, the entire document is checked for valid UTF-8 encoding. • When **`off`**, only the first few bytes are checked for proper encoding. The rest of the document is assumed to be in the same encoding. The default value is **`off`**.	boolean

For examples, see parse.

Proxy

Apply the Proxy policy to invoke another API within your assembly, particularly if the separate API contains a large payload. The response from the backend is stored in the **`message.body`** and in the response object variable if it is defined. Only one policy is permitted to be run per unique assembly flow.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by Invoke	

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see proxy.

About

Only one Proxy policy is permitted to be run per unique flow of your assembly. More than one Proxy policy can be applied, if they are contained in mutually exclusive branches of the assembly.

You can use the Proxy policy to return multipart form data, that is, when the response is set to `Content-Type: multipart/related`. However, Proxy must be the last policy in the assembly, otherwise the response that is received can be manipulated during subsequent steps, thereby causing the multipart form data to be lost.

The proxy policy, if inside a conditional policy, must be the **final** policy to be executed in the API. If you need further processing afterward, use the invoke policy rather than the proxy policy.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Proxy policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`proxy`**.	string
Description	No	A description of the policy.	string
Invoke URL	Yes	Specifies a URL for the target service. For a SOAP API, a URL is added by default. Where possible, the Proxy URL value is pre-supplied from information that is defined in the imported WSDL.	string
TLS profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	Yes	The time to wait before a reply back from the endpoint (in seconds). The default value is **`60`**.	integer
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
HTTP Method	Yes	The HTTP method to use for the proxy. Valid values are: • Keep • GET • POST • PUT • DELETE • PATCH • HEAD • OPTIONS The default value is **`Keep`**. By using **`Keep`**, or removing the property from the source, the HTTP method from the incoming request is used.	string
Compression	No	Select this check box to enable Content-Encoding compression on upload. The check box is cleared by default.	boolean

Property label	Required	Description	Data type
Cache Type	No	The cache type determines whether to cache documents, honoring or overriding the HTTP Cache Control directives received in the response from the target URL. This property takes effect only when a response is received, otherwise the policy always returns the non-expired response that was previously saved in cache. Valid values are: Protocol The cache behavior is determined by the Cache-Control headers on the response, in accordance with RFC 7234. To optimize performance, if the gateway receives more than one request for a resource that is not in the cache but could be cached when the response from the target URL is received, the gateway sends only one request to the target URL; the remaining requests are not processed until the response from the first request has been received and the cache behavior has been determined from this response. If the response indicates that caching is possible, the gateway responds to all waiting requests with the cached resource. If the response indicates that caching is not possible, the gateway sends all waiting requests to the target URL. Use this option only if you expect that responses from the target URL can be cached, in which case it should improve performance and limit the demand on the target URL. If, however, the target URL never indicates that the gateway should cache its response, performance might be impaired when compared to the No Cache option. No Cache Responses from the target URL are not cached on the gateway regardless of any caching headers returned. In this case, every request from the client is sent to the target URL. Use this option if you do not want to cache any of the backend responses on the gateway, or if it is unlikely that a response from the target URL will allow caching through the Cache-Control header settings. Time to Live This option is similar to the Protocol option except it allows you to specify the amount of time that you want the successful response from the invoke or proxy to remain in the cache. Use this option only if you expect that responses from the target URL can be cached. The default value is Protocol.	string
Time to Live	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property Cache type is set to **`Time to Live`**. Enter a value in the range 5 - 31708800. The default value is **`900`**.	integer
Cache key	No	Specifies the unique identifier of the document cache entry. If omitted, the entire URL string is used as the key.	string
Stop on error	No	Select the errors that, if thrown during the policy execution, cause the assembly flow to stop. If there is a **`catch`** flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors selected for the Stop on error setting, or if the error thrown is not one of the selected errors, the policy execution is allowed to complete, and the assembly flow continues.	string
Response object variable	No	The name of a variable that will be used to store the response data from the request. This variable can then be referenced in other actions, such as 'Map'.	string
X-Forwarded header	No	This header can be provided by 1. If **`X-Forwarded-Host`** exists, processing continues. If it does not exist prior to calling the proxy policy, it is set with the value of the **`Host`** header. 2. The **`X-Forwarded-For`** header is always set, in all cases. This header maintains breadcrumbs, showing a comma-separated list of IPs, from the client through any preceding proxy. 3. If all three of **`X-Forwarded-Host`** , **`X-Forwarded-Port`** , and **`X-Forwarded-Proto`** headers are missing at the time of calling the proxy policy, they are set automatically. To prevent this, set **`X-Forwarded-Host`** header to some value before calling the proxy policy.	string

Related tasks

- Creating a new REST OpenAPI definition

Related information

- TLS profiles

API Gateway only

Rate Limit

Use the Rate Limit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.0 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.5.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see ratelimit.

About

The defined rate, burst, and count limits are applied to whatever follows in the assembly flow. For example, if a Rate Limit policy is placed before an Invoke policy, and the call made by the Invoke policy exceeds the limits defined by the Rate Limit policy, the API call itself fails.

Note: For information about rate limits and burst limits in API Connect, see Understanding rate limits for APIs and Plans.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Rate Limit policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`ratelimit`**.	string
Description	No	A description of the policy.	string
Source	Yes	The location of all the rate limit, burst limit, and count limit definitions that are included in this policy. Select one of the following options: • Catalog by Name: the limits to be applied are defined in the appropriate **`api-collection`** object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. For details of how to configure a rate, burst, or count limit in the **`api-collection`** object, see Configuring a rate, burst, or count limit on the DataPower API Gateway. • Plan by Name: the limits to be applied are assembly burst limits or assembly count limits defined on the Plan to which the calling application is subscribed. For details on how to configure assembly burst limits and assembly count limits in a Plan, see Editing a draft Product. Note: The following conditions apply to a Plan by Name rate limit policy: ◦ Every Plan that contains the API that is being limited must have the named rate limit applied. ◦ A Plan by Name rate limit policy cannot be used with automatically generated Products. • Gateway by Name: the limits to be applied are defined in the **`apigw`** object, named **`apiconnect`**, on the DataPower API Gateway gateway. For details of how to configure a rate, burst, or count limit in the **`apigw`** object, see Configuring a rate, burst, or count limit on the DataPower API Gateway. • Plan Default: the rate and burst limits that are applied are the default ones configured in the Plan to which the calling application is subscribed. For details on how to configure default Plan rate and burst limits, see Editing a draft Product. Note: The use of this option does not disable any automatic preflow rate limit policy that might be in place, and could result in the application of the rate limiting being repeated. For more information on preflow policies, see Customizing the preflow policies.	string
Rate Limit Name	Yes[1]	If Source property is set to Catalog by Name, the name of a rate limit as defined in the DataPower API Gateway configuration, To display the Rate Limit Name field, click Add Rate Limit; you can add as many rate limits as you want. [1]You must provide at least one rate, burst, or count limit.	string

Property label	Required	Description	Data type
Burst Limit Name	Yes[2]	If Source property is set to Catalog by Name, the name of a burst limit as defined in the DataPower API Gateway configuration. If Source is set to Plan by Name, the name of an assembly burst limit in a Plan. To display the Burst Limit Name field, click Add Burst Limit; you can add as many burst limits as you want. [2]You must provide at least one rate, burst, or count limit.	string
Count Limit Name	Yes[3]	If Source property is set to Catalog by Name, the name of a count limit as defined in the DataPower API Gateway configuration. If Source is set to Plan by Name, the name of an assembly count limit in a Plan. To display the Count Limit Name field, click Add Count Limit; you can add as many count limits as you want. [3]You must provide at least one rate, burst, or count limit.	string
Operation	No	For a rate limit, select one of the following options: • Consume: the policy reduces the balance of available requests that remain in the interval defined by the rate limit specified in the Rate Limit Name field. The amount by which the remaining balance is reduced is the value that results from the weight expression defined in the specified rate limit. • Replenish: the policy increases the balance of available requests that remain in the interval defined by the rate limit specified in the Rate Limit Name field. The amount by which the remaining balance is increased is the value that results from the weight expression defined in the specified rate limit. The purpose of the replenish operation is to restore an appropriate amount of remaining balance in cases where the amount calculated by an earlier consume operation turns out to be too high. For example, for an assembly rate limit with a limit of 100, a consume operation with a weight of 60 uses 60 requests, leaving 40 requests remaining for the interval. If the weight expression in the subsequent replenish operation computes to 15, meaning that the weight expression calculated that the actual request usage was only 45, the replenish operation increases the remaining requests by 15, to a remaining balance of 55. Note that the replenish operation cannot increase the remaining requests allowed beyond the original limit defined in the assembly rate limit scheme. In this example, a replenish operation cannot replenish requests beyond 100. A replenish operation is skipped if the weight expression evaluates to a value of less than 1. The default value is Consume. For a count limit, select one of the following options: • Increment: the current count is incremented by the value that results from the weight expression defined in the count limit specified in the Count Limit Name field. • Decrement: the current count is decremented by the value that results from the weight expression defined in the count limit specified in the Count Limit Name field. By default, the count limit is automatically decremented at the end of the API assembly by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly. However, if the auto decrement feature has been explicitly disabled in the count limit configuration, you must include a corresponding decrement operation, otherwise when the count limit is reached all future calls are blocked. For more information on configuring a count limit, see Configuring a rate, burst, or count limit on the DataPower API Gateway. The default value is Increment. For a burst limit, the only option is Consume, meaning that the limit applied is that defined in the burst limit specified in the Burst Limit Name field.	string

- **Configuring a rate, burst, or count limit on the DataPower API Gateway**
  If you want to include a rate limiting policy in your API assembly flow, you must first configure the required rate, burst, and count limits on the Gateway. Rate and burst limits restrict the number of calls that an application can make to API in specified time period, while count limits impose a strict limit on the total number of calls.

Configuring a rate, burst, or count limit on the DataPower API Gateway

If you want to include a rate limiting policy in your API assembly flow, you must first configure the required rate, burst, and count limits on the Gateway. Rate and burst limits restrict the number of calls that an application can make to API in specified time period, while count limits impose a strict limit on the total number of calls.

About this task

You configure rate, burst, and count limits on the DataPower® API Gateway by defining them in one or more configuration files that you package and then add to the Gateway as a Gateway extension.

You define *Catalog scoped* rate, burst, and count limits in the appropriate **api-collection** object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. The defined rate, burst, and count limits will be available to be included in Rate Limit policies in the assembly flows of all the APIs that are published to that Catalog.

You define *Gateway scoped* rate, burst, and count limits in the appropriate **apigw** object on the DataPower API Gateway. The defined rate, burst, and count limits will be available to be included in Rate Limit policies in the assembly flows of all the APIs that are published to that Gateway.

Note:

- Only one Gateway extension can be in use at any one time, so you should package all required rate, burst, and count limits in a single Gateway extension.
- To apply a Catalog scoped or Gateway scoped rate, burst, or count limit, it must be referenced in a Rate Limit policy in an API assembly.
- You should ensure that all rate, burst, and count limit names are unique across all your Plans and Catalogs. When an API is called, information about any rate, burst, or count limits (such as name, limiting criteria, and remaining number of calls) is returned in the HTTP headers of the response. If two or more limit definitions with exactly the same name are applied during the invocation of an API, information about only **one** of them is returned. However, **all** limiting criteria are applied.

Procedure

1. Create a .cfg file that includes the DataPower API Gateway CLI commands that define the rate, burst, or count limit.
   To define a rate limit, use the **assembly-rate-limit** command. To define a burst limit, use the **assembly-burst-limit** command. To define a count limit, use the **assembly-count-limit** command. For details of the command parameters, see assembly-rate-limit, assembly-burst-limit, and assembly-count-limit in the DataPower product documentation.

   Example 1: Configuring a Catalog scoped rate limit

   ```
   top; config;
   switch apiconnect;

   api-collection myorg_mycatalog_collection
     assembly-rate-limit  30per1min 30 1 minute on off on on off off "" "my.vars.expected-amount - my.vars.actual-
   amount"
   exit
   ```

   In this example, the **assembly-rate-limit** command specifies a rate limit name, and the API call limits that you want to impose, 30 calls per minute in this case. The final, optional argument, **"my.vars.expected-amount - my.vars.actual-amount"**, is a weight expression. You use weight expressions in cases where you want to restore an appropriate amount of remaining rate limit balance when an earlier usage calculation in your assembly flow turns out to be too high. You define two rate limiting policies in your assembly flow, one with a **consume** operation, and a subsequent one with a **replenish** operation that restores some of the remaining rate limit balance if appropriate. The weight expression calculates the amounts to be consumed or replenished; for more information, see Rate Limit.

   The **api-collection** command specifies the **api-collection** object in which you want to configure the rate, burst, or count limits. The name of the **api-collection** object has the following format:

   ***org_catalog*_collection**

   where:
   - *org* is the name of the provider organization that contains your Catalog.
   - *catalog* is the name of the Catalog.

   This **api-collection** object name is assigned automatically by the Gateway the first time that a Product is published to the Catalog.

   Example 2: Configuring a Catalog scoped count limit

   ```
   top; config;
   switch apiconnect;

   api-collection myorg_mycatalog_collection
     assembly-count-limit  maxsixty 60 1 on on off off off off "" "1" off
   exit
   ```

   The final argument is the **autodecrement** property that controls whether the count limit is automatically decremented at the end of an API assembly. The default value is **on**, meaning that at the end of the assembly the count limit is automatically decremented by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly.

   Example 3: Configuring a Gateway scoped rate limit

   ```
   top; config;
   switch apiconnect;

   apigw apigw_object_name
     assembly-rate-limit  30per1min 30 1 minute on off on on off off "" "my.vars.expected-amount - my.vars.actual-
   amount"
   exit
   ```

   The **apigw** command specifies the name of the **apigw** object in which you want to configure the rate, burst, or count limits; the **apigw** object name is the same as the name of the API Connect domain in your DataPower API Gateway configuration.

2. Package the .cfg file in a **.zip** file.
   You can package multiple .cfg files in a single **.zip** file if you want; the .cfg files must all be at the root of the **.zip** file.
3. Add the **.zip** file to the DataPower API Gateway as a Gateway extension. For details, see Configuring your Gateway server extensions.

What to do next

Apply the rate, burst, or count limits to an API by adding a Rate Limit policy to the API assembly flow.

Related information

- Gateway extension guidelines - DataPower API Gateway.

API Gateway

Redaction - DataPower API Gateway

Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

Gateway support

Note: This page describes the Redaction policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see Redaction - DataPower Gateway (v5 compatible).
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see redact - DataPower API Gateway.

Note: With the DataPower API Gateway, the input to the Redaction policy must be parsed data. One way to produce parsed data is to use a Parse policy before a Redaction policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Redaction policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`redact`**.	string
Description	No	A description of the policy.	string
Root	No	Specifies the data source that contains the content to redact or remove. If no value is entered in the Root field, the action is applied to the entire API context. You can use any supported JSONata path expression. If you want to apply the action to either request or response data, specify a value of `message.body`. The actual content to which the action is applied then depends on the positioning of the Redaction policy in the overall assembly flow; for example: • If positioned at the beginning, the action is applied to the client request. • If positioned after an Invoke policy, the action is applied to the response from the back end. • If positioned at the end, the action is applied to the response that is returned to the client. If, in your assembly flow, the Redaction policy is used after a Log policy that specifies Gather-only mode, specify a Root value of `log.request_body` for the logged request payload, or `log.response_body` for the logged response payload.	string
Path	Yes	Specifies a JSONata path expression that identifies the content to redact or remove from the source. For more information, see Constructing JSONata expressions to redact fields	string
Action	No	Specifies whether you want to remove or redact the content. Choose one of the following options: • Remove: Completely removes the specified fields. • Redact: Redacts (obfuscates with "*"s) the fields to block out the data. The default value is Redact. Note: If a numerical value is being redacted, the redacted value is depicted as `******` and the type is changed to **`string`**.	string

Tip: You can optionally click Add action to specify JSONata expressions for additional fields that you want to remove or redact from the specified content source.

- **Constructing JSONata expressions to redact fields**
  To define a field for redaction or removal when using the Redaction policy with the DataPower API Gateway, you supply a JSONata expression that defines the path to the field that you want to redact or remove.

API Gateway only

Constructing JSONata expressions to redact fields

To define a field for redaction or removal when using the Redaction policy with the DataPower® API Gateway, you supply a JSONata expression that defines the path to the field that you want to redact or remove.

Note: For a list of known limitations to DataPower API Gateway support for JSONata, see Limitations to support for JSONata.
The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`
    In addition to the standard values returned by `$type()`, the API gateway implementation of `$type()` can return the following values: `binary`, `empty`, `graphql`, `stream`, or `xml`.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`
  - `$substring(str, start[, length])`
  - `$substringAfter(str, chars)`
  - `$substringBefore(str, chars)`
  - `$trim(str)`
  - `$uppercase(str)`
- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description

`$apiCtx()`	Generic access to an API context	The `$apiCtx()` extension gives generic access to an API context.
		○ Sample transform field in an extract action:
		`"$apiCtx().request.uri"`
		○ Sample condition of a case in a switch action:
		`"$apiCtx().request.path = '/simple/apictx-function'"`

`$header(name)`	`message.headers.name`	Message header
`$httpVerb()`	`request.verb`	HTTP method of the request
`$operationID()`	`api.operation.id`	ID of the operation
`$operationPath()`	`api.operation.path`	Path of the operation
`$queryParameter('name')`	○ `request.parameters.`*`name`*`.locations` The supported keyword is `query`. ○ `request.parameters.`*`name`*`.values`	Searches for the index of `query` in `request.parameters.`*`name`*`.locations` and returns `request.parameters.`*`name`*`.values[index]`, where `[index]` is the value for `query` in locations. Parameter values are not URL decoded.
`$statusCode()`	`message.status.code`	Status code

Extension	Variable	Description
`$storageType([arg])`	***variable*`.body`** You can specify any variable in the API context. When no variable is specified, the default variable `message.body` is used.	Storage type of the message. The supported values are `binary`, `empty`, `graphql`, `json`, `stream`, or `xml`.

`$urlParameter('name')`	○ `request.parameters.`***name*`.locations`**
	The supported keywords are `path` and `query`
	○ `request.parameters.`***name*`.values`
		Searches for the index of `path` and `query` in `request.parameters.`***name*`.locations` and returns a single array that contains both `path` and `query` values from `request.parameters.`***name*`.values`. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.
		For example, the following URL contains both path and query parameter values.
		`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`
		The `$urlParameter('breed')` URL returns the following array of values.
		`[cats, adopt, Sphynx, Siamese]`
		In this example, the URL includes an API path that is configured as `/petstore/{breed}/{breed}`, where `breed` is configured to be a path parameter of the API path. As a result, `cats` and `adopt` are included in the output.

`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The `xpathExpression` must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. `$xpath($, '//price')`

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
`$gqlActiveOperation([graphql_message])`	`message.body`	Gets the active operation found in the specified GraphQL message. The `operationName` must be the same as the name of the active operation.
`$gqlAlias(graphql_field_node)`	`message.body`	Gets the alias of a GraphQL field node.
`$gqlFragments([graphql_message])`	`message.body`	Gets the fragments found in the specified GraphQL message.
`$gqlName([graphql_node])`	`message.body`	Gets the node name. For operations, the node name is the `operationName`. For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the `operationName` of `message.body` is retrieved.
`$gqlOperations([graphql_message])`	`message.body`	Gets the operations found in the specified GraphQL message.
`$gqlType([graphql_node])`	`message.body`	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- / (division)
- % (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`
  The entire GraphQL query including operations and fragments.

- ***`operationName`***
  For an anonymous operation, ***`operationName`*** can be empty.

- `~`***`fragmentSpreadName`***
- `on~`***`typeCondition`***
- `~~`***`fragmentDefinitionName`***

You can define the path in either of the following ways:

- Use a JSONata expression

- Use the `$xpath()` JSONata extension

Using a JSONata expression

The path specified by the JSONata expression is relative to any value specified for the **root** property of the Redaction policy. If the **root** property has no value or is absent, begin the expression with the absolute content path. If the **root** property has a value then you can either begin the expression with **$** to use the **root** path directly, or you can provide a sub-path relative to the **root** path.

JSONata expressions can be used with content that is in either JSON or XML format.

Example 1
: If the **root** property of the Redaction policy has no value or is absent, use the following expression to redact or remove all occurrences of the **price** field in the request and response data:

```
message.body.**.price
```

Example 2
: If the **root** property of the Redaction policy has the value **log.request_body**, use the following expression to redact or remove all occurrences of the **price** field, specifically within an **item** element, in the logged request payload:

```
$.item.price
```

Example 3
: If the **root** property of the Redaction policy has the value **log**, use the following expression to redact or remove all occurrences of the **price** field in the logged response payload:

```
response_body.**.price
```

The ****** descendant wildcard traverses all descendants at all hierarchical levels.

Using the `$xpath()` JSONata extension

The **$xpath()** function has the following format:

```
$xpath(content_path, xpath_expression)
```

where:

- *content_path* is the path to the content that contains the field that you want to redact or remove.
- *xpath_expression* is the XPath expression that defines the field that you want to redact or remove.

The *content_path* is relative to any value specified for the **root** property of the Redaction policy. If the **root** property has no value or is absent, provide the absolute content path. If the **root** property has a value then you can either provide the value **$** for the *content_path* parameter to use the **root** path directly, or you can provide a sub-path relative to the **root** path.

The **$xpath()** function can be used only with content that is in XML format.

Example 1
: If the **root** property of the Redaction policy has no value or is absent, use the following expression to redact or remove all occurrences of the **price** field in the request and response data:

```
$xpath(message.body, '//price')
```

Example 2
: If the **root** property of the Redaction policy has the value **log.request_body**, use the following expression to redact or remove all occurrences of the **price** field, specifically within an **item** element, in the logged request payload:

```
$xpath($, 'item/price')
```

Example 3
: If the **root** property of the Redaction policy has the value **log**, use the following expression to redact or remove all occurrences of the **price** field in the logged response payload:

```
$xpath(response_body, '//price')
```

The **//** expression in the second parameter selects all occurrences anywhere in the content.

DataPower Gateway (v5 compatible)

Redaction - DataPower Gateway (v5 compatible)

Use the Redaction policy to completely remove or to redact specified fields from the Request body, the Response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

Gateway support

Note: This topic describes the Redaction policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see Redaction - DataPower API Gateway.
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways

support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see redact - DataPower Gateway (v5 compatible).

About

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Redaction policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **redact**.	string
Description	No	A description of the policy.	string
Path	Yes	Specifies an XPath expression that defines the field to remove or redact. You can construct an XPath expression that is based on JSON or XML depending on whether your API requests and responses use a JSON or an XML format. If the payload is JSON, use the DataPower XML representation of the JSON content (JSONx) to construct the expression. Note: Use a JSONx representation only to identify the XPath expressions for the fields to remove or redact. Do not change the format of any response bodies in API Manager. To learn more about constructing XPath expressions that are based on JSON or XML, see Constructing XPath expressions to redact fields.	string
Action	Yes	Specifies whether you want to remove or redact the field. Valid values: • **remove**: Completely removes the specified field. • **redact**: Redacts (obfuscates with "*"s) the field to block out the data. The default value is **redact**. Note: If a numerical value is being redacted, the redacted value is depicted as ****** and the type is changed to **string**.	string
From	Yes	Specifies where to remove or redact the specified field from. Valid values: • **all**: Removes or redacts the specified field from the Request body, the Response body, and the activity logs. • **request**: Removes or redacts the specified field from the Request body. • **response**: Removes or redacts the specified field from the Response body. • **logs**: Removes or redacts the specified field from the activity logs. The default value is **all**. Optionally click Add item to specify additional values.	string

Tip: You can optionally click Add item to specify XPath expressions for additional fields that you want to remove or redact from the Request body, Response body, and logs.

- **Constructing XPath expressions to redact fields**
  To define a field for redaction when using the DataPower Gateway (v5 compatible), you supply an XPath expression that specifies the field that you want to redact. If your API requests and responses use XML format, you can base your XPath statements directly on the XML content. If your API requests and responses use JSON format, you must use the DataPower XML representation of the JSON content, JSONx, to construct your XPath expression.

Related tasks

- Creating a new REST OpenAPI definition

Related reference

- API Connect context variables

DataPower Gateway (v5 compatible) only

Constructing XPath expressions to redact fields

To define a field for redaction when using the DataPower® Gateway (v5 compatible), you supply an XPath expression that specifies the field that you want to redact. If your API requests and responses use XML format, you can base your XPath statements directly on the XML content. If your API requests and responses use JSON format, you must use the DataPower XML representation of the JSON content, JSONx, to construct your XPath expression.

About this task

The following sections provide examples for constructing XPath expressions to redact a field; examples are provided for API responses in both XML and JSON format:

XPaths for XML

Consider the following example of an XML response:

```
<xml>
  <primaryAddress>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </primaryAddress>
  <secondaryAddress>
    <streetAddress>412 Brooklyn Avenue</streetAddress>
    <city>New Jersey</city>
    <state>NJ</state>
    <postalCode>12302</postalCode>
  </secondaryAddress>
</xml>
```

To redact "streetAddress" in the preceding example the XPath expression would be: `//streetAddress`
where the components of the expression have the following meaning:

Expression	Meaning
`//`	search anywhere in the XML structure
`//streetAddress`	search anywhere in the XML structure for an element of type "streetAddress"

The XML response that results from applying this XPath expression is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
  <primaryAddress>
    <streetAddress>*******</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </primaryAddress>
  <secondaryAddress>
    <streetAddress>*******</streetAddress>
    <city>New Jersey</city>
    <state>NJ</state>
    <postalCode>12302</postalCode>
  </secondaryAddress>
</xml>
```

Note: Both incidences of "streetAddress" have been redacted.
To redact one specific incidence of "streetAddress" in the initial example, the XPath expression would be: `//secondaryAddress/streetAddress`
where the components of the expression have the following meaning:

Expression	Meaning
`//secondaryAddress`	search anywhere in the XML structure for an element of type "secondaryAddress"
`//secondaryAddress/streetAddress`	search under the preceding element for a child element of type "streetAddress"

The XML response that results from applying this XPath expression is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
  <primaryAddress>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </primaryAddress>
  <secondaryAddress>
    <streetAddress>*******</streetAddress>
    <city>New Jersey</city>
    <state>NJ</state>
    <postalCode>12302</postalCode>
  </secondaryAddress>
</xml>
```

Note: Only the secondary address has been redacted.
XPaths for JSON
Consider the following example of a JSON response:

```
{
  "primaryAddress": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "secondaryAddress": {
    "streetAddress": "412 Brooklyn Avenue",
    "city": "New Jersey",
    "state": "NJ",
    "postalCode": 12302
  }
}
```

Unlike the preceding XML example, to construct an XPath for this example you must use the corresponding DataPower XML representation, JSONx. The JSONx equivalent of this JSON response is as follows:

```
<json:object xsi:schemaLocation="http://www.datapower.com/schemas/json jsonx.xsd"
xmlns:json="http://www.ibm.com/xmlns/prod/2009/jsonx" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <json:object name="primaryAddress">
    <json:string name="streetAddress">21 2nd Street</json:string>
    <json:string name="city">New York</json:string>
    <json:string name="state">NY</json:string>
    <json:number name="postalCode">10021</json:number>
  </json:object>
  <json:object name="secondaryAddress">
    <json:string name="streetAddress">412 Brooklyn Avenue</json:string>
    <json:string name="city">New York</json:string>
    <json:string name="state">NY</json:string>
    <json:number name="postalCode">10021</json:number>
  </json:object>
</json:object>]
```

To redact the element "streetAddress" from the preceding JSONx structure, the XPath expression would be: `//*[@name='streetAddress']`
where the components of the expression have the following meaning:

Expression	Meaning
`//*`	find any element anywhere in the structure
`[@name='streetAddress']`	this element has a 'name' property with value "streetAddress"

The JSON response that results from applying this XPath expression is as follows:

```
{
  "primaryAddress": {
    "streetAddress": "*******",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "secondaryAddress": {
    "streetAddress": "*******",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  }
}
```

Note: All incidences of "streetAddress" have been redacted in the preceding JSON structure.
To redact a specific occurrence of the "streetAddress" element, the XPath expression would be: `//*[@name='secondaryAddress']/*[@name='streetAddress']`
where the components of the expression have the following meaning:

Expression	Meaning
`//*`	find any element anywhere in the structure
`//*[@name='secondaryAddress']`	find an element anywhere in the structure that is named "secondaryAddress"
`//*[@name='secondaryAddress']/*[@name='streetAddress']`	find a child element of any type (`/*`) where the child element has a name of "streetAddress"

The JSON response that results from applying this XPath expression is as follows:

```
{
  "primaryAddress": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "secondaryAddress": {
    "streetAddress": "*******",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  }
}
```

Note: Only the street address for the secondary address has been redacted.

What to do next

See Including elements in your OpenAPI 2.0 API assembly, Including elements in your OpenAPI 2.0 API assembly to learn how to apply a redact policy using XPath.

Set Variable

Use the Set Variable policy to set the value of a runtime variable, or to clear a runtime variable, or to add a header variable.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see set-variable.

- **Configuring the Set Variable policy for DataPower API Gateway**
  Follow these steps to configure the Set Variable policy for DataPower API Gateway in the assembly user interface.
- **Configuring the Set Variable policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the Set Variable policy for DataPower Gateway (v5 compatible) in the assembly user interface.

Related tasks

- Creating a new REST OpenAPI definition

Related reference

- API Connect context variables

API Gateway only

Configuring the Set Variable policy for DataPower API Gateway

Follow these steps to configure the Set Variable policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the Set Variable policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the Set Variable policy for DataPower Gateway (v5 compatible). For more information about the different types of gateway, see API Connect gateway types. For information about how to configure the policy in your OpenAPI source, see set-variable.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Set Variable policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Set Variable policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`set-variable`**.	string
Description	No	A description of the policy.	string
Action	Yes	Defines what action to apply on a runtime variable. Choose one of the following values: • Set: Indicates that you want to set a runtime variable to a string value. Can be used to set new headers or to override existing values. • Add: Indicates that you want to add a header variable. Can be used only to set new headers or to append a new entry of the same header name. • Clear: Indicates that you want to delete a runtime variable. Can be used to remove a header when the data is processed in the assembly flow. The default value is Set.	string
Set, Add, or Clear	Yes	Specifies the name of the variable that you want to set , add or clear, depending on the selected Action.	string
Type	Yes	Select the data type of the variable. Choose one of the following values: • any • string • number • boolean For all values other than any, the value is validated against the specified data type.	string
Value	Yes*	Allocates this value to the specified variable. Can be a literal value, or another variable. * Value is required only when **`Set`** or **`Add`** is specified as the action. For example, to *set* a named variable of **`billing-hostname`** to a literal value, you can specify the Value as **`acme.com`**. As another example, to *set* a named variable to the value of the Content-Type header in a request, you can specify the Value entry as **`$(request.headers.content-type)`**. If the selected value of the Type field is boolean, select the Value check box to indicated a value of **`true`**. Note: You can only set single string elements. Values are retrieved as strings and therefore you cannot clone a complete nodeset.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **`version`** section of the policy YAML. For example:

```
execute:
  - set-variable:
      version: 2.0.0
      title: set-variable
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

Example

For examples, see set-variable.

DataPower Gateway (v5 compatible) only

Configuring the Set Variable policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Set Variable policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the Set Variable policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the Set Variable policy for DataPower API Gateway. For more information about the different types of gateway, see API Connect gateway types.
For details on how to configure the policy in your OpenAPI source, see set-variable.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Set Variable policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Set Variable policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`set-variable`**.	string
Description	No	A description of the policy.	string
Action	Yes	Defines what action to apply on a runtime variable. Choose one of the following values: • Set: Indicates that you want to set a runtime variable to a string value. Can be used to set new headers or to override existing values. • Add: Indicates that you want to add a header variable. Can be used only to set new headers or to append a new entry of the same header name. • Clear: Indicates that you want to delete a runtime variable. Can be used to remove a header when the data is processed in the assembly flow. The default value is Set.	string
Set, Add, or Clear	Yes	Specifies the name of the variable that you want to set , add or clear, depending on the selected Action.	string
Type	Yes	Select the data type of the variable. Choose one of the following values: • string • number • boolean The value is validated against the specified data type.	string
Value	Yes*	Allocates this value to the specified variable. Can be a literal value, or another variable. * Value is required only when **`Set`** or **`Add`** is specified as the action. For example, to *set* a named variable of **`billing-hostname`** to a literal value, you can specify the Value as **`acme.com`**. As another example, to *set* a named variable to the value of the Content-Type header in a request, you can specify the Value entry as **`$(request.headers.content-type)`**. If the selected value of the Type field is boolean, select the Value check box to indicated a value of **`true`**. Note: You can only set single string elements. Values are retrieved as strings and therefore you cannot clone a complete nodeset.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **`version`** section of the policy YAML. For example:

```
execute:
  - set-variable:
      version: 1.0.0
      title: set-variable
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

Example

For examples, see set-variable.

User Security

Use the User Security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see user-security.

About

When you define an assembly user security action, you can define the processing for identity-extraction, authentication, and authorization or you can selectively disable any of these this aspects of processing. When disabled, this processing aspect is skipped.

When identity-extraction is enabled, the following methods are supported.

- Use basic authentication, which requires no additional configuration.
- Use context variables. For this method, specify which variable contains the user name and password.
- Use a redirect. For this method, specify the URL fragment to redirect to, and the time allowed to process.
- Use an HTML login form. For this method, specify whether to use the default or custom form and the time allowed to submit the form. For a custom form, specify the location of the form and the TLS client profile to secure the connection to the remote server.

When authentication is enabled, the following methods are supported.

- Contact an LDAP server. For this method, specify which server to contact.
- Send a request to an authentication endpoint. For this method, specify the URL of the endpoint, the TLS client profile to secure the connection, the pattern to select which response header to add, and the response header that contains the authenticated credentials.

When authorization is enabled, the following methods are supported.

- Implicitly accept any previously authenticated users, which requires no additional configuration.
- Use an HTML authorization form. For this method, specify whether to use the default or custom form and the time allowed to submit the form. For a custom form, specify the location of the form and the TLS client profile to secure the connection to the remote server.

You can attach this policy to the REST API flow.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. User Security policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`user-security`**.	string
Description	No	A description of the policy.	string
Factor ID	No	The identity that identifies the results of factor-authentication in the API context.	string

Property label	Required	Description	Data type
Extract Identity Settings	Yes	Select the method that is used to extract the user credentials. The following options are available: Basic Use basic authentication; no additional configuration is required. Context Variable The credentials are provided by API Connect context variables; specify the following properties: • Username content variable: the context variable that is used to obtain the user name. • password context variable: the context variable that is used to obtain the password. HTML Form Use forms based identity-extraction. Select whether to use the default form or a custom form. For a custom form, specify the following properties: • Custom form endpoint: the location of the form. • Custom form TLS profile: the TLS client profile that is used to secure the connection to the remote server. In the HTML form time limit field, specify the time allowed to submit the form. Redirect Use a redirect for identity-extraction; specify the following properties: • Redirect URL: the URL fragment to which to redirect the request to obtain user credentials. • Redirect time limit: the time allowed for the transaction to complete. Disabled Identity-extraction is disabled; this aspect of processing is skipped. Select Stop on error to halt assembly processing in the event of identity-extraction failure.	string
Authenticate User Settings	Yes	Select the authentication method. The following options are available: Authentication URL The credentials are authenticated by an external endpoint; specify the following properties: • Authentication URL: the URL of the authentication endpoint. • Authentication TLS profile: the TLS client profile that is used to secure the connection to the authentication endpoint. • Authentication response header pattern: the pattern that is used to select which response headers to add to the API context. • Authentication response header credential: the response header that contains the authenticated user credentials. LDAP The credentials are authenticated by an LDAP user registry; from the LDAP registry list, select the required registry. Disabled Authentication is disabled; this aspect of processing is skipped. Select Stop on error to halt assembly processing in the event of authentication failure.	string
Authorize User Settings	Yes	Select the authorization method. The following options are available: authenticated Implicitly accept any previously authenticated users; no additional configuration is required. HTML Form The user provides authorization through an HTML form. Select whether to use the default form or a custom form. For a custom form, specify the following properties: • Custom form endpoint: the location of the form. • Custom form TLS profile: the TLS client profile that is used to secure the connection to the remote server. In the Dynamic table entries field, enter the name of a context variable that specifies the scopes that are to be added automatically to the authorization consent form. In the HTML form time limit field, specify the time allowed to submit the form. Disabled Authorization is disabled; this aspect of processing is skipped. Select Stop on error to halt assembly processing in the event of authorization failure.	string

Related tasks

- Creating a new REST OpenAPI definition

Related reference

- API Connect context variables

Validate - DataPower API Gateway

Use the Validate policy to validate the payload in an assembly flow against a schema.

Gateway support

Note: This page describes the Validate policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see Validate - DataPower Gateway (v5 compatible).
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.0 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.4.0 (DataPower API Gateway Version 10.0.4.0 or later) 2.5.0 (DataPower API Gateway Version 10.5.0.0 or later) 2.6.0 (DataPower API Gateway Version 10.5.0.2 or later 2.7.0 (DataPower API Gateway Version 10.5.0.3 or later

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see validate - DataPower API Gateway.

About

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate policy after the task that collates the response.

Note: With the DataPower API Gateway, the input to the Validate policy must be parsed data. One way to produce parsed data is to use a Parse policy before a Validate policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **validate**.	string
Description	No	A description of the policy.	string
Input	No	Specifies the name of a variable in the API context. The content of the **body** field of the variable, which is represented by ***variable_name*.body**, is the input data to validate. By default, the variable name is **message**.	string
Output	No	Specifies the name of a variable in the API context. • If the validation passes, the body field of the output variable, which is represented by ***variable_name*.body**, stores the output of the assembly validate action. • If the schema to validate is a JSON schema, the validation also adds any default values that are missing from the payload. • If the validation fails, no output is stored. • If an output variable is not specified, the results of the assembly validate action are not saved.	string

Property label	Required	Description	Data type

Validate against	Yes	Specifies the schema to be used for validating the payload.
		Select one of the following values:
		
		• **`definition`**: Select this value if you want to use a previously defined schema to validate the payload that is returned from other invoke actions or tasks in the assembly flow.
		From the Definition Object list, select the required schema. For information on defining a schema for an OpenAPI 2.0 API definition, see Defining schema definitions for an API. For information on defining a schema for an OpenAPI 3.0 API definition, see Defining schema components.
		
		• **`URL`**: the schema is identified by a URL location.
		  ◦ In the JSON Schema URL field, enter the URL of the JSON schema to be used for validating a JSON payload.
		  ◦ From the XML Validation Mode list, select how XML validation is to be performed:
		  ▪ **`None`**: no XML payload validation is required.
		  ▪ **`xsd`**: an XML schema will be used to validate an XML payload. In the XML Schema URL field, enter the URL of the XML schema.
		  ▪ **`wsdl`**: a WSDL schema will be used to validate a SOAP payload. In the WSDL URL field, enter the URL of the WSDL schema.
		  ▪ **`soap-body`**: validate the body of a SOAP message against the XML schema only.
		  Note: The following limitations apply to schemas used for JSON validation, which include JSON schemas and OpenAPI documents that are used as schemas for validation. Exceeding these limits can impact compilation performance and is not supported.
		  ▪ Maximum of 6,500 lines. Each key and each item in an array count as a line.
		  ▪ Maximum recursion depth of 250.
		  ▪ Maximum of 3,000 items in **`enum`** lists.
		• **`WSDL`**: (available with a SOAP service based API only) use the XML schema in the WSDL file associated with the API operation or the API path.
		• **`body-param`**: validate the request input against the schema definition that is specified in the TYPE field for the request parameter for this operation. For information about how to create a request parameter, see Defining parameters for an operation.
		• **`response-param`**: validate the response to be returned to the client application, against the schema definition that is specified in the SCHEMA field for the response parameter for this operation. For information about how to create a response parameter, see Defining responses for an operation.
		• **`GraphQL`**: (available for a GraphQL proxy API only) validate the payload against the GraphQL schema that has been imported into the GraphQL proxy API. In addition, either the GraphQL query or response, depending on the input, is analyzed against the GraphQL schema to calculate the cost, and the result is placed in the API context.
		For more information on GraphQL proxy APIs, see Creating a GraphQL proxy API and GraphQL context variables.
			string

XSLT version	No	The XSLT processor version. The default value is XSLT10.	string
Strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
Profile rule	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
Debug rule	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
Streaming rule	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
Attempt streaming rule	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
Minimum output escaping rule	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
Maximum stack size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
WS-I Basic Profile validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn. Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string

Property label	Required	Description	Data type
Validate message body	No	The validation behavior for the **soap:Body**. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
Validate message headers	No	The validation behavior for the **soap:Header**. The default setting is Lax. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
Validate message fault details	No	Specifies the validation behavior for the fault detail. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
Require wrappers on fault details specified by type	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
Specifically allow xsi:type='SOAP-ENC:Array' rule	No	Whether to allow the schema to accept most uses of elements with **xsi:type='SOAP-ENC:Array'** consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the **xsi:type** attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, **xsi:type** is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from **SOAP-ENC:Array**. The opposite is the normal allowable case. By default, elements with **xsi:type='SOAP-ENC:Array'** are not accepted.	boolean
Validate SOAP 1.1 encoding rule	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
Wildcards ignore xsi:type rule	No	Whether **xs:any** elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an **xsi:type** attribute on it. This option ignores those **xsi:type** attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, **xsi:type** attributes are not ignored.	boolean
Strict SOAP envelope version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
Debug XACML policy	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
Accept MTOM/XOP optimized messages	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an **xop:Include** reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled. • When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as **xs:base64Binary** or **xs:string**, but the message contains an **xop:Include** element instead. • When enabled, an **xop:Include** element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The **xop:Include** element itself will be validated according to the built-in schema in **store:///schemas/xop.xsd**.	boolean

For examples, see validate - DataPower API Gateway.

Related tasks

- Defining Paths for a REST API
- Creating a new REST OpenAPI definition

DataPower Gateway (v5 compatible)

Validate - DataPower Gateway (v5 compatible)

Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.

Gateway support

Note: This topic describes the Validate policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see Validate - DataPower API Gateway.
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see validate - DataPower Gateway (v5 compatible).

Restriction:

- The schema that represents the XML can reference only one XML namespace.
- The schema cannot reference polymorphic XML elements.
- The validation works on the `message.body` variable and not any other output/context variable. If the invoke policy contains a configured response object variable, then `message.body` is not set, and validate is not able to act.
- If you use the `multipleOf` keyword in a schema definition for the API then, due to rounding behavior, the specified value must satisfy the following conditions, otherwise the validation fails when the API is called:
  - The value must not be less than `0.00000099999999999999848869`.
  - If the value is greater than 1, the amount before the decimal point must not be greater than `99999999999999934463`.

About

You can attach this policy to the following API flow:

- REST

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate policy after the task that collates the response.

You can apply a different OpenAPI schema definition to each Validate policy either by choosing from the set of schema definitions that is specified at the API level, or the schema definition at the operation level.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 2. Validate policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is `validate`.	string
Description	No	A description of the policy.	string
Definition	Yes	Specifies the schema definition to be used for validating the payload. Valid values: • `request`: Select this value to validate the request input against the schema definition that is specified in the TYPE field for the request parameter for this operation. For information about how to create a request parameter, see Configuring an operation. • `response`: Select this value to validate the response to be returned to the client application, against the schema definition that is specified in the SCHEMA field for the response parameter for this operation. For information about how to create a response parameter, see Configuring an operation. • `#/definitions/definition_name`: Select this value for a previously defined schema definition to be used to validate the payload that is returned from other invoke actions or tasks in the assembly flow. For information on defining a schema for an API definition, see Editing an API definition.	string

Related tasks

- Defining Paths for a REST API
- Creating a new REST OpenAPI definition

Validate Username Token

Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0 1.1.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see validate-usernametoken.

About

A WS-Security UsernameToken enables a user identity to be passed securely over a multi-point message path. The Validate Username Token policy extracts the UsernameToken element from the request payload, authenticates the extracted username and password, and provides access to the protected resource based on the authentication result. The policy has two authentication methods: Lightweight Directory Access Protocol (LDAP) user registry, or Authentication URL.

The Validate Username Token policy supports both passwordText and passwordDigest types of password. When the authentication method is Authentication URL with a passwordDigest, a basic authentication header that contains a Base64 encoded username and passwordDigest is sent to the URL. In addition, a custom header named X-IBM-PasswordType is set with a value of digest. The following table shows the authentication process based on password type:

Table 2. Authentication URL process by password type

passwordText	passwordDigest
Authentication: Basic base64(username:password)	Authentication: Basic base64(username:passwordDigest) X-IBM-PasswordType: 'digest'

Position this policy where required in the assembly flow as follows:

- To validate the original input, position a Validate Username Token policy at the start of your flow.
- To validate an intermediate response that is returned from other invoke actions or tasks, position a Validate Username Token policy after those actions or tasks.
- To validate the response that is returned to the client application, position a Validate Username Token policy after the task that collates the response.

Properties

The following table lists the policy properties, indicates whether a property is required, specifies the valid and default values for input, and specifies the data type of the values.

Table 3. Validate Username Token policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`validate-usernametoken`**.	string
Description	No	A description of the policy.	string
Authentication type (policy version 1.0.0 only)	Yes	The authentication type to use to validate the UsernameToken. Valid values: • **`Authentication URL`**: Select this value to validate the user credentials against an authentication URL. • **`LDAP registry`**: Select this value to validate the user credentials against an LDAP user registry. The default value is: **`Authentication URL`**.	string
Authentication URL (policy version 1.0.0 only)	Yes	The authentication URL to use to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to **`Authentication URL`**.	string
TLS profile (policy version 1.0.0 only)	No	The TLS profile to use for the secure transmission of data to the authentication URL. Note: This property is available only if Authentication type is set to **`Authentication URL`**.	string
LDAP registry name (policy version 1.0.0 only)	Yes	The name of the LDAP user registry to validate the UsernameToken user credentials against. You can select a name from the drop-down list, or type a name manually. Note: This property is required only if Authentication type is set to **`LDAP registry`**.	string
User Registry Name (policy version 1.1.0 and later)	Yes	Select the LDAP or Authentication URL registry to use to validate the UsernameToken.	string
LDAP search attribute[1]	Yes	The name of the LDAP user password attribute. Note: This property is required only for an LDAP user registry.	string

Examples

The following example shows an LDAP user registry authentication:

```
- validate-usernametoken:
    version: 1.0.0
    title: "validate-usernametoken"
    auth-type: "LDAP Registry"
    ldap-registry: "wstest"
    ldap-search-attribute: "userPassword"
```

The following example shows an Authentication URL definition:

```
- validate-usernametoken:
    version: 1.0.0
    title: "validate-usernametoken"
    auth-type: "Authentication URL"
    auth-url: "https://www.google.com"
    tls-profile: "default-ssl-profile"
```

Errors

The policy returns an HTTP `200` status code when successful, and the input payload is copied to the output flow. For all failure types the policy returns an HTTP `500` status code, and the output contains the SOAP fault.

Tip: If there are authentication failures, try verifying the LDAP user registry configuration as follows:

- Ensure Search (DN) is set as the communication method.
- Ensure Authenticated Bind is set so that specific permissions are required to search the registry.
- Ensure the Admin DN and Password fields are correctly completed for the Distinguished Name (DN) of a user authorized to carry out searches in the LDAP directory.
- Ensure that a combination of Base DN, Prefix, and Suffix are set, such that they fully describe the user DN. For example:
  - For a user named: **`cn=alice`**, **`dc=ibm`**, **`dc=com`**

    ```
    BaseDN: dc=ibm
    Prefix: cn=alice
    Suffix: dc=com
    ```

    where the user DN is calculated as: Prefix + BaseDN + Suffix.

Related concepts

- LDAP authentication

Related tasks

- Creating a new REST OpenAPI definition

Related information

- Creating an LDAP registry
- Authenticating by using your enterprise user registry

[1] When authenticating with LDAP and passwordText, the policy uses the username and password as LDAP bind credentials. However, when authenticating with LDAP and passwordDigest, the digest itself cannot be used for authentication. Instead, an LDAP search for the username is performed by using the administrator's distinguished name (DN) and password, and an attribute corresponding to the contents of the ldap-search-attribute is retrieved. A hash of the contents of this attribute (along with the Nonce and Created attributes, as in the WS-Security UsernameToken profile specification) is then compared to the passwordDigest.

Websocket Upgrade

Use the Websocket Upgrade policy to process API requests and responses through a WebSocket connection.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

- **Configuring the Websocket Upgrade policy for DataPower API Gateway**
  Follow these steps to configure the Websocket Upgrade policy for DataPower API Gateway in the assembly user interface.
- **Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the Websocket Upgrade policy for DataPower Gateway (v5 compatible) in the assembly user interface.

API Gateway only

Configuring the Websocket Upgrade policy for DataPower API Gateway

Follow these steps to configure the Websocket Upgrade policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the Websocket Upgrade policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible). For more information about the different types of gateway, see API Connect gateway types.
For information about how to configure the policy in your OpenAPI source, see websocket-upgrade.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Websocket Upgrade policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. Websocket Upgrade policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`websocket-upgrade`**.	string
Description	No	A description of the policy.	string
URL	Yes	Specify the URL to be invoked.	string
TLS Profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is **`60`**.	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code **`301 Moved Permanently`**. If you select this check box, the **`invoke`** policy follows the URL redirection by making a further call to the URL specified in the **`Location`** header in the response. If you clear this check box, the **`invoke`** saves the **`301`** status code and the API call is considered to be complete. Note: The **`follow-redirect`** property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the **`invoke`** always follows the URL redirection; the **`proxy`** policy (not supported by the DataPower API Gateway) saves the **`301`** status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
Inject proxy headers	No	If you select this check box, the **`invoke`** policy injects the **`X-Forwarded-For`**, **`X-Forwarded-To`**, **`X-Forwarded-Host`**, and **`X-Forwarded-Proto`** headers to the request that is sent to the target URL. The check box is cleared by default.	boolean
Decode Request Params	No	If you select this check box, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The check box is cleared by default.	boolean
Queryparam encode	No	If you select this check box, all "+" characters in the query parameter values of the target URL are encoded to %2F. The check box is cleared by default.	boolean
Header control	No	Specifies the headers in **`message.headers`** that you want to copy to the target URL. To prevent headers from being copied, complete the following steps: a. Select Blocklist. b. Click Add blocklist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. To specify headers that you want to be copied, complete the following steps: a. Select Allowlist. b. Click Add allowlist. c. In the empty field that is displayed, enter the header name. d. To add further headers, repeat the previous steps. The values that you specify are in regular expression format. For example, to specify the Content-Type header, enter **`^Content-Type$`** By default, Blocklist is selected, with no blocklist entries, meaning that **all** headers are copied.	string

Property label	Required	Description	Data type

Parameter control	No	Specifies the parameters in the incoming request that you want to be copied to the target URL.
		To prevent parameters from being copied, complete the following steps:
		a. Select Blocklist.
		b. Click Add blocklist.
		c. In the empty field that is displayed, enter the parameter name.
		d. To add further parameters, repeat the previous steps.
		To specify parameters that you want to be copied, complete the following steps:
		a. Select Allowlist.
		b. Click Add allowlist.
		c. In the empty field that is displayed, enter the parameter name.
		d. To add further parameters, repeat the previous steps.
		The values that you specify are in regular expression format.
		
		For example, if the incoming request is
		
		`http://apigw/org/sandbox/petstore/base?petid=100&display=detailed`
		
		and you specify a white list entry of `^petid$`, the target URL at run time will be
		
		`http://myhost/mypath?storeid=3&petid=100`
		
		By default, Allowlist is selected, with no allowlist entries, meaning that **no** parameters are copied.
			string

Request assembly	No	The request processing assembly. For details on configuring an assembly in the `execute` section, see execute. For example:
		
		`request-assembly:`
		`  execute`
		`  .`
		`  .`
		`  .`
		`  policy assembly`
		`  .`
		`  .`
		`  .`
			object

Response assembly	No	The response processing assembly. For details on configuring an assembly in the `execute` section, see execute. For example:
		
		`response-assembly:`
		`  execute`
		`  .`
		`  .`
		`  .`
		`  policy assembly`
		`  .`
		`  .`
		`  .`
			object

6. Specify a version for the policy by clicking the Source icon </>, and completing the `version` section of the policy YAML. For example:

```
execute:
  - websocket-upgrade:
      version: 2.0.0
      title: websocket-upgrade
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

DataPower Gateway (v5 compatible) only

Configuring the Websocket Upgrade policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the Websocket Upgrade policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the Websocket Upgrade policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the Websocket Upgrade policy for DataPower API Gateway. For more information about the different types of gateway, see API Connect gateway types. For details on how to configure the policy in your OpenAPI source, see websocket-upgrade.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the Websocket Upgrade policy in the palette, and drag the policy onto your canvas.

5. Specify the following properties.

Table 1. Websocket Upgrade policy properties

Property label	Required	Description	Data type
Title	No	The title of the policy. The default value is **`websocket-upgrade`**.	string
Description	No	A description of the policy.	string
URL	Yes	Specify the URL to be invoked.	string
TLS Profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
Timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is **`60`**.	integer
Follow redirects	No	Specifies the behavior if the back-end server returns the HTTP status code **`301 Moved Permanently`**. If you select this check box, the **`invoke`** policy follows the URL redirection by making a further call to the URL specified in the **`Location`** header in the response. If you clear this check box, the **`invoke`** saves the **`301`** status code and the API call is considered to be complete. Note: The **`follow-redirect`** property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the **`invoke`** always follows the URL redirection; the **`proxy`** policy (not supported by the DataPower API Gateway) saves the **`301`** status code and completes the API call without following the URL redirection.	boolean
Username	No	The username to use for HTTP Basic authentication.	string
Password	No	The password to use for HTTP Basic authentication.	string
Inject proxy headers	No	If you select this check box, the **`invoke`** policy injects the **`X-Forwarded-For`**, **`X-Forwarded-To`**, **`X-Forwarded-Host`**, and **`X-Forwarded-Proto`** headers to the request that is sent to the target URL. The check box is cleared by default.	boolean
Decode Request Params	No	If you select this check box, any request parameters that are referenced by a variable definition on the target URL of the invoke policy are URL-decoded. The check box is cleared by default.	boolean
Queryparam encode	No	If you select this check box, all "+" characters in the query parameter values of the target URL are encoded to %2F. The check box is cleared by default.	boolean

Request assembly	No	The request processing assembly. For details on configuring an assembly in the **`execute`** section, see execute. For example:
		`request-assembly:`
		`  execute`
		`  .`
		`  .`
		`  .`
		`  policy assembly`
		`  .`
		`  .`
		`  .`
			object

Response assembly	No	The response processing assembly. For details on configuring an assembly in the **`execute`** section, see execute. For example:
		`response-assembly:`
		`  execute`
		`  .`
		`  .`
		`  .`
		`  policy assembly`
		`  .`
		`  .`
		`  .`
			object

6. Specify a version for the policy by clicking the Source icon </>, and completing the **`version`** section of the policy YAML. For example:

```
execute:
  - websocket-upgrade:
      version: 1.0.0
      title: websocket-upgrade
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

XML to JSON

Use the XML to JSON policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

The XML to JSON policy uses a simple convention, based on BadgerFish, to convert your API context payload from XML to JSON. The XML content is preserved, including the attributes and namespaces. No additional configuration is required. For more information about the BadgerFish convention, including some examples, see BadgerFish.

Use the API Designer assembly view when you are creating your API definition to add a built-in policy to the flow.

The policy must be attached to the flow at the point at which you require the conversion to be performed. For example, if you need to convert an XML-formatted request into a JSON-formatted request, the policy must be attached to the request flow.

The policy reads input from the **message.body**, if that context exists, otherwise from the **request.body**, and then writes the output to the **message.body**.

Note: If you are using the DataPower API Gateway, the input to the XML to JSON policy must be parsed data. One way to produce parsed data is to use a Parse policy before an XML to JSON policy in your assembly flow, which provides explicit control of the parse action.

Examples

For example, the following simple XML object

```
<a>hello</a>
```

becomes

```
{ "a": { "$" : "hello" } }
```

The following XML object with an attribute

```
<a type="world">hello</a>
```

becomes

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

For examples, see xml-to-json.

- **Configuring the XML to JSON policy for DataPower API Gateway**
  Follow these steps to configure the XML to JSON policy for DataPower API Gateway in the assembly user interface.
- **Configuring the XML to JSON policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the XML to JSON policy for DataPower Gateway (v5 compatible) in the assembly user interface.

API Gateway only

Configuring the XML to JSON policy for DataPower API Gateway

Follow these steps to configure the XML to JSON policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the XML to JSON policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the Set Variable policy for DataPower Gateway (v5 compatible). For more information about the different types of gateway, see API Connect gateway types. This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see xml-to-json.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the XML to JSON policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XML to JSON policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **xml-to-json**.	string
Description	No	A description of the policy.	string
Input	No	The input message to convert. Specify the name of a variable in the API context. ***variableName*.body**, the message payload, represents the XML input to convert. The default value of the variable is **message** and **message.body** is the default input.	string
Output	No	The output message to store the conversion result. Specify the name of a variable in the API context. ***variableName*.body** represents the result of conversion from XML format to JSON format. When the specified input message is the default message, the default output is **message.body**. Otherwise, when the input message is the variable **my-message-variable**, for example, the default output is **my-message-variable.body**. The variable cannot be any read-only in the API context.	string
Conversion type	No	The conversion type that determines the target format of the output. The following options are available: • badgerFish: BadgerFish convention is used to determine the target conversion format of the output. • apicv5: apicv5 convention is used to determine the target conversion format of the output.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **version** section of the policy YAML. For example:

```
execute:
  - xml-to-json:
      version: 2.0.0
      title: xml-to-json
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

Example

For example, the following simple XML object

```
<a>hello</a>
```

becomes

```
{ "a": { "$" : "hello" } }
```

The following XML object with an attribute

```
<a type="world">hello</a>
```

becomes

```
{ "a": { "$" : "hello", "@type" : "world" } }
```

For more examples, see xml-to-json.

DataPower Gateway (v5 compatible) only

Configuring the XML to JSON policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the XML to JSON policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the XML to JSON policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the XML to JSON policy for DataPower API Gateway. For more information about the different types of gateway, see API Connect gateway types.
This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see xml-to-json.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the XML to JSON policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XML to JSON policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **xml-to-json**.	string
Description	No	A description of the policy.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **version** section of the policy YAML. For example:

```
execute:
  - xml-to-json:
      version: 1.0.0
      title: xml-to-json
  ...
```

You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.

7. Click Save.

Example

For examples, see xml-to-json.

XSLT

Use the XSLT policy to apply an XSLT transform to the payload of the API definition.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.4.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see xslt.

Note: If you are using the DataPower API Gateway, the input to the XSLT policy must be parsed data. One way to produce parsed data is to use a Parse policy before an XSLT policy in your assembly flow, which provides explicit control of the parse action.
For examples of the OpenAPI definitions of XSLT policies, see XSLT policy examples.

For more examples of how to use XSLT to access and modify properties and context, see Implementation code examples and, if you are using the DataPower API Gateway, Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

Errors

The following error can be thrown while the policy is being executed:

- **`TransformError`** - a generic error that captures all errors that occur during the execution of the policy.

- **Configuring the XSLT policy for DataPower API Gateway**
  Follow these steps to configure the XSLT policy for DataPower API Gateway in the assembly user interface.
- **Configuring the XSLT policy for DataPower Gateway (v5 compatible)**
  Follow these steps to configure the XSLT policy for DataPower Gateway (v5 compatible) in the assembly user interface.
- **XSLT policy examples**
  Examples of the OpenAPI definitions of XSLT policies.
- **Implementation code examples**
  Example XSLT and GatewayScript code snippets.

Related concepts

- Variable references in API Connect

Related tasks

- Creating a new REST OpenAPI definition

Related reference

- API Connect context variables

API Gateway only

Configuring the XSLT policy for DataPower API Gateway

Follow these steps to configure the XSLT policy for DataPower® API Gateway in the assembly user interface.

About this task

Note: This topic describes the XSLT policy implementation in the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), see Configuring the XSLT policy for DataPower Gateway (v5 compatible). For more information about the different types of gateway, see API Connect gateway types.
For information about how to configure the policy in your OpenAPI source, see xslt.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the XSLT policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XSLT policy properties

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **`xslt`**.	string
Description	No	A description of the policy.	string

Property label	Required	Description	Data type
Input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The check box is cleared by default, which indicates that there is no input.	boolean
Serialize output	No	If you select this option, the output tree that is generated by the XSLT policy is serialized. The content of **`message.body`** is updated with the serialized binary data rather than the XML tree. The check box is cleared by default, which indicates that the output tree is not serialized.	boolean
Source	Yes	The XSLT transform source to execute.	string
XSLT version	No	The XSLT processor version. The default value is XSLT10.	string
Strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
Profile rule	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
Debug rule	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
Streaming rule	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
Attempt streaming rule	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
Minimum output escaping rule	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
Maximum stack size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
WS-I Basic Profile validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn. Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string
Validate message body	No	The validation behavior for the **`soap:Body`**. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
Validate message headers	No	The validation behavior for the **`soap:Header`**. The default setting is Lax. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
Validate message fault details	No	Specifies the validation behavior for the fault detail. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string

Property label	Required	Description	Data type
Require wrappers on fault details specified by type	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
Specifically allow xsi:type='SOAP-ENC:Array' rule	No	Whether to allow the schema to accept most uses of elements with `xsi:type='SOAP-ENC:Array'` consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the `xsi:type` attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, `xsi:type` is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from `SOAP-ENC:Array`. The opposite is the normal allowable case. By default, elements with `xsi:type='SOAP-ENC:Array'` are not accepted.	boolean
Validate SOAP 1.1 encoding rule	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
Wildcards ignore xsi:type rule	No	Whether `xs:any` elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an `xsi:type` attribute on it. This option ignores those `xsi:type` attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, `xsi:type` attributes are not ignored.	boolean
Strict SOAP envelope version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
Debug XACML policy	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
Accept MTOM/XOP optimized messages	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an `xop:Include` reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled. • When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as `xs:base64Binary` or `xs:string`, but the message contains an `xop:Include` element instead. • When enabled, an `xop:Include` element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The `xop:Include` element itself will be validated according to the built-in schema in `store:///schemas/xop.xsd`.	boolean

6. Specify a version for the policy by clicking the Source icon </>, and completing the `version` section of the policy YAML. For example:

```
execute:
  - xslt:
      version: 2.1.0
      title: xslt
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

DataPower Gateway (v5 compatible) only

Configuring the XSLT policy for DataPower Gateway (v5 compatible)

Follow these steps to configure the XSLT policy for DataPower® Gateway (v5 compatible) in the assembly user interface.

About this task

Note: This topic describes the XSLT policy implementation in the DataPower Gateway (v5 compatible). If you are using the DataPower API Gateway, see Configuring the XSLT policy for DataPower API Gateway. For more information about the different types of gateway, see API Connect gateway types.
For details on how to configure the policy in your OpenAPI source, see xslt.

Procedure

1. In the navigation pane, click Develop, then select the APIs tab.
   The Develop page opens.
2. Click the title of the API that you want to work with, or create a new API.
3. Select the Gateway tab, then click Policies in the navigation pane.
   For more information about working with the assembly editor for an API, see The assembly editor.
4. Find the XSLT policy in the palette, and drag the policy onto your canvas.
5. Specify the following properties.

Table 1. XSLT policy properties

Property label	Required	Description	Data type

Property label	Required	Description	Data type
Title	Yes	The title of the policy. The default value is **xslt**.	string
Description	No	A description of the policy.	string
Input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The check box is cleared by default, which indicates that there is no input.	boolean
Source	Yes	The XSLT transform source to execute.	string

6. Specify a version for the policy by clicking the Source icon </>, and completing the **version** section of the policy YAML. For example:

```
execute:
  - xslt:
      version: 1.0.0
      title: xslt
  ...
```

   You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.
7. Click Save.

XSLT policy examples

Examples of the OpenAPI definitions of XSLT policies.

- Simple example with no context current payload
- Concatenation and transformation
- Obtain query parameter values and refer to context variables
- Return DataPower object name associated with a client TLS profile

Attention:

- The mechanisms described here for interacting with the API Connect context are intended for v5-compatible gateways and also to provide v5 compatibility for APIs created for the API Gateway. However, if you are writing an XSLT policy or user-defined policy for a new API Gateway API, use the mechanisms described in Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway for better performance and long-term support. For API Gateway APIs, the v5-compatibility functions listed here are to be used for v5 migration only.
- The XML specification https://www.w3.org/TR/xml/ does not specify a preferred order for XML namespace (XMLNS) attributes. Best practice is to not rely upon the sequence of XMLNS attributes if you write custom parsing code.

Simple example with no context current payload

The following is an example of where the XSLT input document does not use the context current payload (there is no input):

```
- xslt:
  title: example xslt
  source: |
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="/">
        <Hello>World!</Hello>
      </xsl:template>
    </xsl:stylesheet>
```

Concatenation and transformation

The following example shows a more complex XSLT transform source, where the stylesheet concatenates two input strings and transforms the third input string to the IP address of the client:

```
- xslt:
  title: xslt
  input: true
  source: |
    <?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xalan="http://xml.apache.org/xslt"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:dp="http://www.datapower.com/extensions"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs4xs="http://www.w3.org/2001/XMLSchema"
        xmlns:io="http://xformMessage"
        xmlns:map="http://xformMessage/xform"
        xmlns:msl="http://www.ibm.com/xmlmap"
        exclude-result-prefixes="fn dp dp map xalan msl"
        version="1.0">
      <xsl:output method="xml" encoding="UTF-8" indent="no"/>

      <!-- root wrapper template  -->
      <xsl:template match="/">
        <msl:datamap>
          <xsl:choose>
            <xsl:when test="not(msl:datamap/dataObject[1]/@xsi:nil)">
              <xsl:element name="dataObject">
                <xsl:attribute name="xsi:type">
                  <xsl:value-of select="'io:data'"/>
                </xsl:attribute>
```

```
              <xsl:call-template name="map:xform">
                <xsl:with-param name="data" select="msl:datamap/dataObject[1]"/>
              </xsl:call-template>
            </xsl:element>
          </xsl:when>
          <xsl:otherwise>
            <xsl:element name="dataObject">
              <xsl:attribute name="xsi:type">
                <xsl:value-of select="'io:data'"/>
              </xsl:attribute>
              <xsl:attribute name="xsi:nil">
                <xsl:text>true</xsl:text>
              </xsl:attribute>
            </xsl:element>
          </xsl:otherwise>
        </xsl:choose>
      </msl:datamap>
    </xsl:template>

    <!-- This rule represents a type mapping: "data" to "io:data".  -->
    <xsl:template name="map:xform">
      <xsl:param name="data"/>
      <!-- a simple data mapping: "$data/StringOne"(string) to "StringOne"(string) -->
      <xsl:if test="$data/StringOne">
        <StringOne>
          <xsl:value-of select="concat($data/StringOne, $data/StringTwo)"/>
        </StringOne>
      </xsl:if>
      <!-- a simple mapping with no associated source:  to "StringTwo"(string) -->
      <StringTwo>
        <xsl:value-of select="dp:client-ip-addr()"/>
      </StringTwo>
      <!-- a simple data mapping: "$data/NumberOne"(int) to "NumberOne"(int) -->
      <xsl:if test="$data/NumberOne">
        <NumberOne>
          <xsl:value-of select="$data/NumberOne"/>
        </NumberOne>
      </xsl:if>
      <!-- a simple data mapping: "$data/NumberTwo"(int) to "NumberTwo"(int) -->
      <xsl:if test="$data/NumberTwo">
        <NumberTwo>
          <xsl:value-of select="$data/NumberTwo"/>
        </NumberTwo>
      </xsl:if>
      <!-- a simple data mapping: "$data/NumberThree"(int) to "NumberThree"(int) -->
      <xsl:if test="$data/NumberThree">
        <NumberThree>
          <xsl:value-of select="$data/NumberThree"/>
        </NumberThree>
      </xsl:if>
    </xsl:template>

    <!-- *****************    Utility Templates    ******************  -->
    <!-- copy the namespace declarations from the source to the target -->
    <xsl:template name="copyNamespaceDeclarations">
      <xsl:param name="root"/>
      <xsl:for-each select="$root/namespace::*[not(name() = '')]">
        <xsl:copy/>
      </xsl:for-each>
    </xsl:template>
  </xsl:stylesheet>
```

Obtain query parameter values and refer to context variables

The following example shows a complete OpenAPI source file. The API includes an XSLT policy that obtains a query parameter value in XSLT, and also uses the **getvariable** method to retrieve the value of the context variable **request.headers.user-agent**.

```
swagger: '2.0'
info:
  x-ibm-name: xslt
  title: xslt
  version: 1.0.0
schemes:
  - https
host: $(catalog.host)
basePath: /xslt
consumes:
  - application/json
produces:
  - application/json
securityDefinitions:
  clientIdHeader:
    type: apiKey
    in: header
    name: X-IBM-Client-Id
security:
  - clientIdHeader: []
x-ibm-configuration:
  testable: true
  enforced: true
  cors:
    enabled: true
  assembly:
    execute:
```

```
    - operation-switch:
        title: operation-switch
        case:
          - operations:
              - verb: get
                path: /hello
            execute:
              - xslt:
                  title: SayHello
                  input: false
                  source: |
                    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
                      <xsl:template match="/">
                        <xsl:element name="APIc">
                           <xsl:text>Hello World!</xsl:text>
                        </xsl:element>
                      </xsl:template>
                    </xsl:stylesheet>
          - operations:
              - verb: get
                path: /getContextQueryVar
            execute:
              - xslt:
                  title: GetContextQueryVar
                  input: false
                  source: |
                    <xsl:stylesheet version="1.0"
                      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                      xmlns:apim="http://www.ibm.com/apimanagement">
                      <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"/>
                      <xsl:template match="/">
                        <xsl:call-template name="apim:output">
                          <xsl:with-param name="mediaType" select="'application/xml'"/>
                        </xsl:call-template>
                        <APIC>
                          <xsl:element name="apim.getVariable">
                            <xsl:element name="useragent">
                              <xsl:value-of select="apim:getVariable('request.headers.user-agent')"/>
                            </xsl:element>
                            <xsl:element name="query">
                              <xsl:value-of select="apim:getVariable('request.querystring')"/>
                            </xsl:element>
                          </xsl:element>
                        </APIC>
                      </xsl:template>
                    </xsl:stylesheet>
          - operations:
              - verb: get
                path: /getQuery
            execute: []
        otherwise:
          - throw: null
            title: handling unknown operation
            name: Unsupported
    catch:
      - errors:
          - Unsupported
        execute:
          - set-variable:
              actions:
                - set: message.body
                  value: '<error>Not Supported</error>'
  phase: realized
paths:
  /hello:
    get:
      responses:
        '200':
          description: 200 OK
  /getContextQueryVar:
    get:
      responses:
        '200':
          description: 200 OK
definitions: {}
tags: []
```

Return DataPower object name associated with a client TLS profile

The following example uses the **apim:getTLSProfileObjName** function to return the DataPower object name associated with a specified client TLS profile.

```
- xslt:
  title: xslt
  input: false
  version: 2.0.0
  source: >-
    <?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:dp="http://www.datapower.com/extensions"
      xmlns:apim="http://www.ibm.com/apimanagement"
      exclude-result-prefixes="dp apim"
      extension-element-prefixes="dp"
```

```
      version="1.0">

      <!-- For apigw ... store:///dp/apim.custom.xsl -->
      <!-- For v5/v5c ... local:///isp/policy/apim.custom.xsl -->
      <!-- <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"/>
-->
      <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl"/>

      <xsl:template match="/">
        <xsl:variable name="httpHeaders">
          <header name="Accept">application/xml</header>
        </xsl:variable>
        <!--
          For v5, the name returned is client:<orgname>-my-test
          For v5c, the name returned is client:<orgname>-my-testV1.0.0
          For API Gateway, the name returned is client:<orgname>_<catalogname>_tlsp-my-testV1.0.0
        -->
        <xsl:variable name="tlsClientProfile" select="apim:getTLSProfileObjName('my-test')" />
        <xsl:variable name="targetUrl" select="'https://127.0.0.1:6201/PingRequest'"/>
        <dp:url-open target="{$targetUrl}" response="responsecode" timeout="20" http-headers="$httpHeaders" ssl-proxy="
{$tlsClientProfile}"/>
      </xsl:template>

    </xsl:stylesheet>
```

Note that references to the "v5c" (v5-compatible) gateway do not apply to API Connect Enterprise as a Service.

For more examples of how to use XSLT to access and modify properties and context, see Implementation code examples and, if you are using the DataPower® API Gateway, Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

Implementation code examples

Example XSLT and GatewayScript code snippets.

Note: If you are using GatewayScript, you must include one or other of the following commands depending on your gateway type:

DataPower Gateway (v5 compatible)
```
var apic = require(./apim.custom.js);
```
API Gateway
```
var apic = require('apim');
```

where *apic* is the common name used for the GatewayScript examples in this topic. However, *apic* could be any given name of your choice, for example you could use:

```
var apim = require(./apim.custom.js);
```

and then you would start your calls with `apim`.

- Access to input properties code snippet
- Access to runtime context code snippet
- Access to input payload code snippet
- Access to HTTP headers code snippet
- Modify the payload code snippet
- Configure error information code snippet
- Accessing the caught exception in a catch block
- Set variables code snippet

Access to input properties code snippet

The following code block shows an example of how to access the input properties by using the XSLT `policyProperties()` function. The example defines a property that is named `a_property`, which is declared as an integer value, but is retrieved in XSLT as text.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
```
DataPower Gateway (v5 compatible)
```
<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
```
API Gateway
```
<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="p" select="apim:policyProperties()" />
    <xsl:message>
      The value of my input property is
      <xsl:value-of select="$p/a_property" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.getPolicyProperty(propertyName)
```

where ***propertyName*** is the name of the input property that you want to access. If the input property name is blank, the action will return all input properties.

Access to runtime context code snippet

The following code block shows an example of how to access the runtime context by using the XSLT **getContext()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
```

```
<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
```

```
<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="client-id" select="apim:getContext('client.app.id')" />
    <xsl:message>
      The calling application is
      <xsl:value-of select="$client-id" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.getContext(varName)
```

where ***varName*** is the name of the context variable that you want to access.
For a complete list of context variables, see API Gateway context variables. If you are using the DataPower® API Gateway, see also Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

Access to input payload code snippet

The following code block shows an example of how to access the input payload by using the XSLT **payloadRead()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
```

```
<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
```

```
<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="input" select="apim:payloadRead()" />
    <xsl:message>
      The input payload is
      <xsl:copy-of select="$input" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.readInput(callback)
```

A callback is required because the actual payload read is asynchronous. The callback method is called when the payload is ready.
This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.
The following example shows how to use the **payloadType()** function to determine what type of payload (XML or JSONx) will be returned by the XSLT **payloadRead()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
```

```
  DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
  API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="payloadType" select="apim:payloadType()" />
    <xsl:message>
      <xsl:text>Payload type is [</xsl:text>
      <xsl:value-of select="$payloadType" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

Access to HTTP headers code snippet

The following code block shows an example of how to access the HTTP headers in XSLT by using the `getContext()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
  API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="content-type" select="apim:getContext('request.headers.content-type')" />
    <xsl:message>
      The request content type is
      <xsl:value-of select="$content-type" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, call the following function:

```
apic.getContext(request.headers.headerName)
```

where ***headerName*** maps to the name of the header you want to access.
Note: Access or modification of HTTP headers by using DataPower extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.

Modify the payload code snippet

The output from a user-defined policy must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. The following code block shows an example of how to modify the payload in XSLT. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template, as shown in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:apim="http://www.ibm.com/apimanagement"
  xmlns:jsonx="http://www.ibm.com/xmlns/prod/2009/jsonx">

  <!-- Contains the APIM functions -->
  DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
  API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <!-- Creates a JSON document (empty object is for simplicity) -->
    <jsonx:object>
    </jsonx:object>

    <!-- Indicates the media type of the output being produced -->

    <xsl:call-template name="apim:output">
      <xsl:with-param name="mediaType" select="'application/json'" />
    </xsl:call-template>
  </xsl:template>

</xsl:stylesheet>
```

where **mediaType**:

- **'application/json'** is when the output is written in JSONx format.
- **'application/xml'** is when the output is written in XML format.

If you are using GatewayScript, call the following function:

```
apic.output(mediaType)
```

where ***mediaType*** is:

- **application/json** is when the output is written in JSONx format.
- **application/xml** is when the output is written in XML format.

Specifying the media type allows the next steps in the assembly flow to understand how to process the new payload.

Tip: The output from a user-defined policy must be XML or JSONx. JSONx is an IBM standard format to represent JSON as XML. One way to convert output GatewayScript JSON data into JSONx, is to add a **Convert Query Params to XML** action to follow the GatewayScript action within the same policy rule. The **Convert Query Params to XML** action must have an **Input Conversion** with the **Default Encoding** set to **JSON**. The output from the GatewayScript action must be the input for the **Convert Query Params to XML** action for JSONx to be produced.

Configure error information code snippet

The following XSLT code block shows an example of how to configure the policy implementation to produce error information by calling the **apim-error** template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:apim="http://www.ibm.com/apimanagement">

  <!-- Contains the APIM functions -->
  DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
  API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <!-- Indicates this policy has a failure and provides
       additional information for the client application -->
  <xsl:template match="/">
    <xsl:call-template name="apim:error">
      <xsl:with-param name="httpCode" select="'401'" />
      <xsl:with-param name="httpReasonPhrase" select="'Unauthorized'" />
      <xsl:with-param name="errorMessage" select="'Please select a Plan'" />
    </xsl:call-template>
  </xsl:template>

</xsl:stylesheet>
```

where:

- **httpCode** is the code of the required error message.
- **httpReasonPhrase** is the reason for the error.
- **errorMessage** is the suggested action for the user.

If you are using GatewayScript, call the following function:

```
apic.error(name, httpCode, httpReasonPhrase, message)
```

where:

- **name** is the name of the error.
- **httpCode** is the code of the required error message.
- **httpReasonPhrase** is the reason for the error.
- **message** is the suggested action for the user.

Accessing the caught exception in a catch block

The following XSLT code block shows an example of how, in the **catch** block of an API assembly, you can obtain the details of the current caught exception. A possible use would be to create a custom error response using the details of the caught exception.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  DataPower Gateway (v5 compatible)xmlns:apim="http://www.ibm.com/apimanagement"
  API Gatewayxmlns:apigw="http://www.ibm.com/xmlns/datapower/2017/11/apigateway"
  extension-element-prefixes="dp"
  DataPower Gateway (v5 compatible)exclude-result-prefixes="dp apim">
  API Gatewayexclude-result-prefixes="dp apim apigw">

  <xsl:output method="xml" />
```

```
  <!-- Contains the APIM functions -->
    DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
    API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">

    DataPower Gateway (v5 compatible)<xsl:variable name="exception" select="apim:getError()"/>
    API Gateway<xsl:variable name="exception" select="apigw:get-error()"/>
    <!-- output desired error message based on the exception -->
    <myError>
       <errorReason><xsl:value-of select="$exception/error/message"/></errorReason>
    </myError>

    <!-- Propagate the HTTP status code and reason phrase from the exception -->
    <xsl:call-template name="apim:setVariable">
      <xsl:with-param name="varName" select="'message.status.code'"/>
      <xsl:with-param name="value" select="$exception/error/status/code"/>
      <xsl:with-param name="action" select="'Set'" />
    </xsl:call-template>

    <xsl:call-template name="apim:setVariable">
      <xsl:with-param name="varName" select="'message.status.reason'"/>
      <xsl:with-param name="value" select="$exception/error/status/reason"/>
      <xsl:with-param name="action" select="'Set'" />
    </xsl:call-template>

    </xsl:template>

</xsl:stylesheet>
```

The `apim:getError()` and `apigw:get-error()` functions return an XML node set; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<error>
    <name>RuntimeError</name>
    <message>This is a thrown Runtime Error</message>
    <policyTitle>Throw Runtime Error</policyTitle>
    <status>
        <code>500</code>
        <reason>Internal Server Error</reason>
    </status>
</error>
```

If you are using GatewayScript, call the following function:

```
apim.getError()
```

which returns a JSON object; for example:

```
{
  "name": "OperationError",
  "message": "This is a thrown Operation Error",
  "policyTitle": "Throw Operation Error",
  "status": {
    "code": "500",
    "reason": "Internal Server Error"
  }
}
```

Set variables code snippet

The following XSLT code block shows an example of how to set a runtime variable to a specified string value by calling the `setVariable` template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
    DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
    API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:call-template name="apim:setVariable">
      <xsl:with-param name="varName" select="'serviceEndpoint'" />
      <xsl:with-param name="value" select="'https://endpoint.host.com/data'" />
    </xsl:call-template>
    <xsl:message>
      <xsl:text>Variable [</xsl:text>
      <xsl:value-of select="'serviceEndpoint'" />
      <xsl:text>] set to [</xsl:text>
```

```
      <xsl:value-of select="'https://endpoint.host.com/data'" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

where:

- **varName** is the name of the runtime variable that you want to set a value to.
- **value** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **value** as **$(request.headers.content-type)**.

If you are using GatewayScript, call the following function:

```
apic.setvariable(varName, varValue, action)
```

where:

- **varName** is the name of the runtime variable that you want to set a value to, or that you want to add or clear.
- **varValue** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **varValue** as **request.headers.content-type**. This property is only required when **set** or **add** is specified as the action.
- **action** is the action that you want to apply to the variable. Valid options are:
  - **set**
  - **add**
  - **clear**

  If no option is set, the default option of **set** is applied.

The following XSLT example shows how to retrieve the value of a runtime variable by using the **getVariable()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  DataPower Gateway (v5 compatible)<xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_local:_isp_policy_apim.custom.xsl"
/>
  API Gateway<xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.toolkit.doc_store:_dp_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="varValue" select="apim:getVariable('serviceEndpoint')" />
    <xsl:message>
      <xsl:text>Variable [</xsl:text>
      <xsl:value-of select="'serviceEndpoint'" />
      <xsl:text>] = [</xsl:text>
      <xsl:value-of select="$varValue" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

where

- **varValue** is the name of the runtime variable that you want to retrieve a value for.

If you are using GatewayScript, call the following function:

```
apic.getvariable(varName)
```

where ***varName*** is the name of the runtime variable that you want to retrieve a value for.

Logic Constructs

IBM® API Connect includes a number of logic constructs that you can use to apply preconfigured logic to an assembly to control the flow of data through your assembly when the API is called.

Logic constructs are configured in the context of an API. You can use the API Designer assembly editor to add a logic construct to an API and to configure the properties for that construct.

The logic constructs are described in the following subtopics:

- **if**
  Use the if construct to apply a section of the assembly when a condition is fulfilled.
- **operation-switch**
  Use the operation-switch component to apply a section of the assembly to a specific operation.
- **switch**
  Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.

- **throw**
  Use the throw policy to throw an error when it is reached during the execution of an assembly flow.

Related concepts

- The assembly editor

if

Use the if construct to apply a section of the assembly when a condition is fulfilled.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by switch	

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see if.

An if construct provides a way to branch an API's assembly when a specified condition is fulfilled. Each if construct contains a section of the assembly that is only executed when the script within the construct returns a **`true`** value.

When using the assemble view's property sheet, use the Condition field to write your condition that returns **`true`** or **`false`**.

If you want one or more policies or constructs to be executed when the condition of the if construct is fulfilled, drag the new policy or construct onto one of the dashed boxes that are displayed within the if construct. Constructs and policies included in the if construct are part of the case that is executed when the condition of the if construct is returned as true.

For information about the OpenAPI implementation of an if construct, see if.

In the Condition field, use the form **`apim.getvariable('`*`context.location.variable`*`')`** to reference your variables, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

Construct property details

You can configure a construct's properties in the property sheet in the assemble view.

Table 2. The properties of an if construct

Property	Required	Description
Title	No	A custom title for your construct when it is displayed in the canvas. If a title is not specified, **`if`** is used by default.
Description	No	A description of your construct, it is not displayed on the canvas.
Condition	Yes	Use GatewayScript to provide conditions. A list of context variables that you can use to generate conditions can be found in API Connect context variables.

operation-switch

Use the operation-switch component to apply a section of the assembly to a specific operation.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see operation-switch.

An operation-switch component provides a way to branch an API's assembly depending on the operation that is called. For each case, a separate piece of the assembly is applied to the operations that belong to the case.

When using the assemble view's property sheet, click the Case field to view suggested operations. Type in the Case field to refine the list of suggested operations.

Each case behaves as an assembly. To add components to a case, drag the component from the palette to the area included in the operation-switch component. Dashed boxes are displayed where the component can be included in the case.

For information about the OpenAPI implementation of an operation-switch component, see operation-switch.

Restriction: Nesting an operation-switch component inside an if or switch construct, or another operation-switch component, is **not** supported.

Component property details

You can configure a component's properties in the property sheet in the assemble view.

Table 2. The properties of an operation-switch component

Property	Required	Description
Title	No	A custom title for your component when it is displayed in the canvas. If a title is not specified, `operation-switch` is used by default.
Description	No	A description of your component, it is not displayed on the canvas.
Case	Yes	Each case includes one or more operations to which the branch provided by the operation switch applies.

switch

Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see switch.

A switch construct provides a way to branch an assembly based on multiple conditions. Each switch component contains multiple cases, each corresponding to a section of the assembly that is only executed when the condition or operation specified by the case is met or used. Additionally, an otherwise case executes when no other case is fulfilled.

Add new cases by clicking + Case and add an "otherwise" case by clicking + Otherwise.

If multiple cases are fulfilled, the highest priority case will be executed. Change the priority of cases by clicking the Move up ↑and Move down ↓ icons.

To configure a case to be executed if a specific operation is called, use the search operations field and select your operation from the list. You can refine the results of the search by typing in the search operations field.

To configure a case to be executed based on a GatewayScript condition, click edit condition and enter your script in the "Condition editor" window. When you have supplied a script, you can edit your script either in the Condition field or by clicking edit condition.

If you are using the DataPower Gateway (v5 compatible), you enter your condition as GatewayScript directly in a code area in the Condition editor. If you are using DataPower API Gateway, the Condition editor provides a script builder to help you construct your condition script; for more information, see Using the switch policy condition editor.

To delete a case, click the Remove case icon .

If you want one or more policies or constructs to be executed when the condition of a case is fulfilled, drag the new policy or construct onto one of the dashed boxes that are displayed within the case's section of the switch construct.

Note: A switch case **must** contain at least one policy, otherwise the Gateway server returns an error.

Construct property details

You can configure a construct's properties in the property sheet in the assemble view.

Table 2. The properties of a switch construct

Property	Required	Description
Title	No	A custom title for your construct when it is displayed in the canvas. If a title is not specified, `switch` is used by default.
Description	No	A description of your construct, it is not displayed on the canvas.
case	Yes (one or more)	Specify one or more operations or write a script for a condition. API Gateway Use the Condition editor to construct your condition script. See Using the switch policy condition editor. DataPower Gateway (v5 compatible) Use GatewayScript to provide conditions. You can reference variables by using the form `apim.getvariable('context.location.variable')`, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable. A list of context variables that you can use to generate conditions can be found in API Connect context variables.
otherwise	No	Add an otherwise case if you want to execute a section of the assembly when no other cases are fulfilled.

- **Using the switch policy condition editor**
  The switch policy condition editor provides a user interface to help you build the conditions for the `case` clauses in a switch policy.

API Gateway only

Using the switch policy condition editor

The switch policy condition editor provides a user interface to help you build the conditions for the **case** clauses in a switch policy.

You build a condition script by using the Add Group and Add Condition options, together with operator selections, and pre-supplied function selections. The Output field shows the resultant condition script, and updates dynamically as you build your condition.

Adding conditions

When you open the policy condition editor, an initial condition is pre-supplied ready for you to configure.

Use the drop-down list to select the function that you want to begin your condition.

The following JSONata functions are supported:

- Aggregation functions:
  - `$average(array)`
  - `$max(array)`
  - `$min(array)`
  - `$sum(array)`
- Array functions:
  - `$append(array1, array2)`
  - `$count(array)`
  - `$reverse(array)`
  - `$sort(array [, function])`
  - `$zip(array1, ...)`
- Boolean functions:
  - `$boolean(arg)`
  - `$exists(arg)`
  - `$not(arg)`
- Numeric functions:
  - `$abs(number)`
  - `$ceil(number)`
  - `$floor(number)`
  - `$formatBase(number [, radix])`
  - `$number(arg)`
  - `$power(base, exponent)`
  - `$round(number [, precision])`
  - `$sqrt(number)`
- Object functions:
  - `$keys(object)`
  - `$lookup(object, key)`
  - `$merge(array<object>)`
  - `$spread(object)`
  - `$type(value)`
    In addition to the standard values returned by **`$type()`**, the API gateway implementation of **`$type()`** can return the following values: **`binary`**, **`empty`**, **`graphql`**, **`stream`**, or **`xml`**.
- String functions:
  - `$contains(str, pattern)`
  - `$join(array[, separator])`
  - `$length(str)`
  - `$lowercase(str)`
  - `$match(str, pattern [, limit])`
  - `$pad(str, width [, char])`
  - `$replace(str, pattern, replacement [, limit])`
  - `$split(str, separator [, limit])`
  - `$string(arg)`
  - `$substring(str, start[, length])`
  - `$substringAfter(str, chars)`
  - `$substringBefore(str, chars)`
  - `$trim(str)`
  - `$uppercase(str)`
- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description

`$apiCtx()`	Generic access to an API context	The **`$apiCtx()`** extension gives generic access to an API context.
		○ Sample transform field in an extract action:
		`"$apiCtx().request.uri"`
		○ Sample condition of a case in a switch action:
		`"$apiCtx().request.path = '/simple/apictx-function'"`

`$header(name)`	`message.headers.name`	Message header
`$httpVerb()`	`request.verb`	HTTP method of the request

Extension	Variable	Description
`$operationID()`	`api.operation.id`	ID of the operation
`$operationPath()`	`api.operation.path`	Path of the operation
`$queryParameter('name')`	○ `request.parameters.`*`name`*`.locations` The supported keyword is `query`. ○ `request.parameters.`*`name`*`.values`	Searches for the index of `query` in `request.parameters.`*`name`*`.locations` and returns `request.parameters.`*`name`*`.values[index]`, where `[index]` is the value for `query` in locations. Parameter values are not URL decoded.
`$statusCode()`	`message.status.code`	Status code
`$storageType([arg])`	*`variable`*`.body` You can specify any variable in the API context. When no variable is specified, the default variable `message.body` is used.	Storage type of the message. The supported values are `binary`, `empty`, `graphql`, `json`, `stream`, or `xml`.

`$urlParameter('name')`	○ `request.parameters.`*`name`*`.locations` The supported keywords are `path` and `query`
	○ `request.parameters.`*`name`*`.values`
		Searches for the index of `path` and `query` in `request.parameters.`*`name`*`.locations` and returns a single array that contains both `path` and `query` values from `request.parameters.`*`name`*`.values`. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.
		For example, the following URL contains both path and query parameter values.
		`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`
		The `$urlParameter('breed')` URL returns the following array of values.
		`[cats, adopt, Sphynx, Siamese]`
		In this example, the URL includes an API path that is configured as `/petstore/{breed}/{breed}`, where `breed` is configured to be a path parameter of the API path. As a result, `cats` and `adopt` are included in the output.

`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The `xpathExpression` must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. `$xpath($, '//price')`

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
`$gqlActiveOperation([graphql_message])`	`message.body`	Gets the active operation found in the specified GraphQL message. The `operationName` must be the same as the name of the active operation.
`$gqlAlias(graphql_field_node)`	`message.body`	Gets the alias of a GraphQL field node.
`$gqlFragments([graphql_message])`	`message.body`	Gets the fragments found in the specified GraphQL message.
`$gqlName([graphql_node])`	`message.body`	Gets the node name. For operations, the node name is the `operationName`. For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the `operationName` of `message.body` is retrieved.
`$gqlOperations([graphql_message])`	`message.body`	Gets the operations found in the specified GraphQL message.
`$gqlType([graphql_node])`	`message.body`	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- / (division)
- % (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- `query`
  The entire GraphQL query including operations and fragments.
- `operationName`
  For an anonymous operation, `operationName` can be empty.
- `~fragmentSpreadName`
- `on~typeCondition`
- `~~fragmentDefinitionName`

You can also use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Extension	Variable	Description
`$header(name)`	`message.headers.name`	Message header
`$httpVerb()`	`request.verb`	HTTP method of the request
`$operationID()`	`api.operation.id`	ID of the operation
`$operationPath()`	`api.operation.path`	Path of the operation
`$queryParameter('name')`	• `request.parameters.name.locations` • `request.parameters.name.values`	Returns the value of a request query parameter given the parameter name. The function searches the `request.parameters.name.locations` array for the index position of the value `query`, and returns `request.parameters.name.values[index]`, where `index` is the index position. Parameter values are not URL decoded.
`$statusCode()`	`message.status.code`	Status code
`$storageType([arg])`	`variable.body` You can specify any variable in the API context. When no variable is specified, the default variable `message.body` is used.	Storage type of the message. The supported values are: • `binary` • `json` • `stream` • `xml`

`$urlParameter('name')`	• `request.parameters.name.locations`
	• `request.parameters.name.values`
		Given a request parameter name, returns the parameter values for all occurrences of that parameter as a path parameter or query parameter.
		The function searches the `request.parameters.name.locations` array for the index positions of the values `path` and `query`, and returns, in a single array, the `request.parameters.name.values[index]` values for all identified index positions. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.
		For example, the following URL contains both path and query parameter values:
		`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`
		The function call `$urlParameter('breed')` returns the following array of values:
		`[cats, adopt, Sphynx, Siamese]`
		In this example, the URL includes an API path that is configured as `/petstore/{breed}/{breed}`, where `breed` is configured to be a path parameter of the API path. As a result, `cats` and `adopt` are included in the output.

`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The `xpathExpression` must be a literal string.	Allows use of XPath expressions

After you select the function, additional fields are displayed, depending on the selected function, so that you can complete the condition. For example, if you select the function `$httpVerb()`, a comparison operator selection list is displayed (`=` or `!=`), together with a list of HTTP verbs to select on the right hand side of the expression (`GET`, `PUT`, `POST`, `DELETE`, `HEAD`, `OPTIONS`, or `PATCH`).

If you select NOT, your condition is negated with a `$not` function.

To add further conditions, click Add Condition; you then select whether to insert an `and` or an `or` operator between this condition and the preceding one.

To group two or more conditions inside parentheses, click Add Group, then add your conditions to the group; you then select whether to insert an `and` or an `or` operator between this group and the preceding condition or group.

To write your own condition from scratch, select Custom, then enter your script in the field provided.

Example

To build the following condition:

```
($statusCode() != 200 and ($httpVerb() = 'GET' or $httpVerb() = 'PUT'))
```

complete these steps in the condition editor:

1. Select `$statusCode()`, select `!=` for the comparison operator, and enter `200` in the field on the right hand side of the condition.
2. Click Add Group.
3. In the new group sub pane that is displayed, click Add Condition.
4. Select `httpVerb()`, leave the comparison operator as `=`, and select `GET` on the right hand side of the condition.
5. Click Add Condition in the group sub pane again.
6. Select `httpVerb()`, leave the comparison operator as `=`, and select `PUT` on the right hand side of the condition.

7. For the comparison operator between the two conditions, select **or**.

Simple condition statements

The following examples show conditions that use a single function.

This example uses the **$httpVerb()** extension to specify the HTTP method of the request.

```
$httpVerb()="GET"
```

This example uses the **$operationPath()** extension to specify the path of the operation.

```
$operationPath()="/base/path-2"
```

This example uses the **$operationID()** extension to specify the operation ID.

```
$operationID()="test-gatewayscript-GET"
```

This example uses the **$statusCode()** extension to specify the message status code.

```
$statusCode()=200
```

This example uses the **$header(name)** extension to specify the content type of the message header.

```
$header("Content-Type")="application/json"
```

Combining conditions with logical operators

You can use **and** and **or** operators to combine multiple functions in a single condition.

This example specifies an HTTP GET request and an API operation path equal to **test-gatewayscript-GET**.

```
$httpVerb()="GET" and $operationPath()="test-gatewayscript-GET"
```

This example specifies either a **POST** or **PUT** request.

```
$httpVerb()="POST" or $httpVerb()="PUT"
```

This example specifies an API operation ID equal to **test-gatewayscript_POST** and a message status code equal to **200**, or an API operation ID equal to **test-gatewayscript-GET** and a message status code equal to **500**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) or ($operationID()="test-gatewayscript-GET" and
$statusCode()=500)
```

This example specifies an API operation ID equal to **test-gatewayscript-POST** and a message status code equal to **200** with an API operation path equal to **/base/path-2**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) and $operationPath()="/base/path-2"
```

This example specifies a message header content type equal to **text/plain** and a message header length equal to **300**, or a message status code equal to **200**.

```
($header("Content-Type")="text/plain" and $header("Content-Length")=300) or $statusCode()= 200
```

throw

Use the throw policy to throw an error when it is reached during the execution of an assembly flow.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in the assembly user interface; for details on how to configure the policy in your OpenAPI source, see throw.

When the throw policy is encountered, the specified error and error message is produced.

If a catch has been configured that the error produced by the throw policy fulfills, the catch will be triggered.

If no catch is triggered by the thrown error, then a `500 Internal Server Error` is returned to the API caller.

Component property details

You can configure a component's properties in the property sheet in the assemble view.

Table 2. The properties of a throw component

Property	Required	Description
Title	No	A custom title for your component when it is displayed in the canvas. If a title is not specified, **throw** is used by default.

Property	Required	Description
Error name	Yes	The error name that is thrown by the policy.
API Gateway only Error status code (policy version 2.1.0 and later)	No	Specify the HTTP status code for the error. You can use the `$(variable)` format to reference the `message.status.code` API context variable.
API Gateway only Error status reason (policy version 2.1.0 and later)	No	Specify the HTTP reason phrase for the error. You can use the `$(variable)` format to reference the `message.status.reason` API context variable.
Error message	No	The error message that is returned with the error name.

API Gateway only

Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway

In the code for GatewayScript and XSLT policies in your API assemblies, you can use API context variables to work with the messages that are generated when an API is called.

All API context variables are stored in a JSON-like tree; each context variable is a node in this tree. A non-leaf node can be either an object or an array, while a leaf node can be a number, string, boolean, or document; a document can contain an XML document, BLOB, Stream, or another JSON document.

Each context variable has a name and a value. You can access a context variable by name when using the GatewayScript and XSLT APIs. Dot notation is used to access a context variable, reflecting the position of the corresponding node in the JSON-like tree; for example, `request.headers.Host`. Where a node name contains special characters, bracket notation is supported; for example, `request.headers['Content-Type']`.

For the GatewayScript policy, all the APIs for the DataPower® API Gateway are in the `context` module, which is a global module. You can therefore access the `context` module directly, you do not need to use the `require()` function first.

For the XSLT policy, the APIs are in the `http://www.ibm.com/xmlns/datapower/2017/11/apigateway` namespace; to use these APIs you must declare this namespace.

Attention: The mechanisms described here for interacting with the API Connect context are preferred and provide the best performance if you are writing new GatewayScript or XSLT policies, or if you are authoring user-defined policies. However, for v5-compatible gateways and v5 compatibility for APIs created for the API Gateway, the preferred mechanisms are as described in GatewayScript code examples and XSLT policy examples.

- Accessing context variables
- About payloads
- Manipulating the message headers
- Manipulating the message status - examples
- Accessing the message body
- Rejecting the session

Accessing context variables

GatewayScript examples

To read the `client.app.id` context variable:

```
var clientID = context.get('client.app.id');
```

To set the value of a context variable:

```
context.set('my.vars.amount', 100);
```

The resulting tree has the following structure:

```
{ "my": { "vars": { "amount": 100 } } }
```

To delete the node `my.vars`, and all its child nodes:

```
context.clear('my.vars');
```

The resulting tree has the following structure:

```
{ "my": { } }
```

XSLT examples

To read the `client.app.id` context variable:

```
<xsl:variable name="clientID" select="apigw:get-variable('client.app.id')"/>
```

To set the value of a context variable:

```
<apigw:set-variable name="'my.vars.amount'" value="100"/>
```

The resulting tree has the following structure:

```
{ "my": { "vars": { "amount": 100 } } }
```

To delete the node `my.vars`, and all its child nodes:

```
<apigw:clear-variable name="'my.vars'"/>
```

The resulting tree has the following structure:

```
{ "my": { } }
```

About payloads

When the assembly flow for an API call begins, the following two message objects are predefined in the API context tree:

- **request**: holds the information for the original payload; the **request** message is a read-only object.
- **message**: holds the information for the current payload.

Typically, a message object has **headers** and **body** properties. For the current payload, the **message** message object might have **variables** and **status** properties, and other properties created during the execution of the assembly.
The following example shows the general structure of a typical message:

```
{
  "headers": {
    "Content-Type": "application/xml",
    "X-Client-ID": "1234567890"
  },
  "variables": {
    "amount": 123,
    "name": "xyz"
  },
  "status": {
    "code": 200,
    "reason": "OK"
  },
  "body": <XML/Blob/Stream/JSON Document>
}
```

Manipulating the message headers

GatewayScript examples

To delete a header:

```
context.message.header.remove('X-My-Header');
```

To read a header:

```
var reqType = context.request.header.get('Content-Type');
var currType = context.message.header.get('Content-Type');
```

To set a header:

```
context.message.header.set('Content-Type', 'application/json');
```

To iterate through the headers:

```
var reqHdrs = context.request.headers;
for (var h in reqHdrs) {
   console.log('>>> Reading the request header: ', h, '=', reqHdrs[h]);
}

var currHdrs = context.message.headers;
for (var h in currHdrs) {
   console.log('>>> Reading the current header: ', h, '=', currHdrs[h]);
}
```

To create a message context called **headersContext**, if one doesn't already exist, and add to it a boolean context variable called **HelloWorld**, with the value **true**:

```
var ctx = context.getMessage("headersContext") || context.createMessage("headersContext");
ctx.setVariable('HelloWorld', true);
```

The **setVariable** call writes the variable into **headersContext.variables.HelloWorld**; therefore a condition statement in any subsequent **switch** policy should check for **headersContext.variables.HelloWorld** rather than **headersContext.HelloWorld**.
To read the context variable in any subsequent **gatewayscript** policy, use the following call:

```
ctx.getVariable("HelloWorld")
```

Note: If you are either migrating an API from IBM® API Connect Version 5.0, or from the DataPower Gateway (v5 compatible), to the DataPower API Gateway, or you are configuring a new API for the DataPower API Gateway, creating a message context, rather than using the context variable mechanism described in Accessing context variables, is the preferred option. In the case of migration, it most closely matches the behavior of IBM API Connect Version 5.0, and **setVariable** and **getVariable** calls will continue to work without modification.
Creating a message context is also the preferred option if you are creating a new API for the DataPower API Gateway because it most closely aligns the API context with the way that various native DataPower API Gateway policies process it.

XSLT examples

To delete a header:

```
<apigw:clear-header message="'message'" name="'X-My-Header'"/>
```

To read a header:

```
<xsl:variable name="reqType" select="apigw:get-header('request', 'Content-Type')"/>
<xsl:variable name="currType" select="apigw:get-header('message', 'Content-Type')"/>
```

To set a header:

```
<apigw:set-header name="'Content-Type'" value="'text/xml'"/>
```

To get all headers:

```
<xsl:variable name="allReqHdrs" select="apigw:all-headers('request')"/>
<xsl:variable name="allCurrHdrs" select="apigw:all-headers('message')"/>
```

Manipulating the message status - examples

Note: Only the `message` object has the status property, not the `request` object.

GatewayScript examples

To read the status code and reason phrase:

```
var code = context.message.statusCode;
var reason = context.message.reasonPhrase;
```

To update the status code:

```
context.message.statusCode = 400; //integer, update the code, with default reason
```

To update the status code and reason phrase in a single operation:

```
context.message.statusCode = '200 Wonderful'; //string, update code along with customized reason
```

Note: The `context.message.reasonPhrase` context variable is read-only.

XSLT examples

To read the status code and reason phrase:

```
<xsl:variable name="code" select="apigw:get-variable('message.status.code')"/>
<xsl:variable name="reason" select="apigw:get-variable('message.status.reason')"/>

<apigw:set-variable name="'message.status.code'" value="200"/>
<apigw:set-variable name="'message.status.reason'" value="'OK'"/>
```

Accessing the message body

Note: The request body is read-only.

GatewayScript examples

To read the request body:

```
context.request.body.readAsJSON(function(error, json) {});
context.request.body.readAsXML(function(error, xml) {});
context.request.body.readAsBuffer(function(error, buffer) {});
```

To read the message body:

```
context.message.body.readAsJSON(function(error, json) {});
context.message.body.readAsXML(function(error, xml) {});
context.message.body.readAsBuffer(function(error, buffer) {});
```

To write the message body and update the content type accordingly:

```
// text response
context.message.body.write("Hello world.");
context.message.header.set('Content-Type', "text/plain");

// JSON response
var pet = { "kind": "dog", "age": 3, "color": "black" };
context.message.body.write(pet);
context.message.header.set('Content-Type', "application/json");

// XML response
var resp = "<order><id>100</id><timestamp>20181231</timestamp><status>fulfilled</status></order>";
context.message.body.write(resp);
context.message.header.set('Content-Type', "text/xml");
```

XSLT examples

Note:

- There is no XSLT API to write the message body; the output of the stylesheet transformation is used to update `message.body`.
- To read the request body or the message body, you must parse `message.body` with a Parse policy before the XSLT policy is executed. At the beginning of the assembly flow, `message.body` contains `request.body` so, after parsing `message.body`, you can then read the request from `message.body`.
- If the XSLT input document uses the context current payload, `message.body` is used for the input document `'/'` that can be accessed from the XSLT policy if `message.body` has been parsed as XML in advance; this is determined by the `input` property of the XSLT policy, see xslt.

Read the message body:

```
<xsl:variable name="currentPayload" select="apigw:read-payload('message')" />
```

Read the message body when the `input` property of the XSLT policy is set to `true`:

```
<xsl:variable name="currentPayload" select="/" />
```

Rejecting the session

Note: After the session is rejected, the API assembly flow is stopped and a fault response is returned, unless a catch handler is defined.

GatewayScript example

To reject the session with an error name and message, and update the status reason and code:

```
context.reject('CustomError', 'You are not authorized to make this API call');
context.message.statusCode = '401 Unauthorized';
```

XSLT example

To reject the session with an error name and message, and update the status reason and code:

```
<apigw:reject identifier="'CustomError'" status-code="401" reason="'Unauthorized'">You are not authorized to make this API call</apigw:reject>
```

- **Manipulating attachments in a message object**
  The message object provides APIs and properties to manipulate message attachments.

Manipulating attachments in a message object

The message object provides APIs and properties to manipulate message attachments.

If the assembly includes a parse policy, attachments in the request or response of a multipart message can be manipulated and sent back to the client. By default, the response is sent to the client in multipart format.
Restriction: You cannot add an attachment to a request or response which does not contain existing attachments.
Alternatively, you can clear `context.message.attachments` to discard the attachments and send only `message.body` to the client.
Note: If the assembly does not include a parse policy, the following processing rules apply.

- For a multipart request where the `message.body` variable is not changed, the invoke policy keeps and passes the attachments to the server.
- For a multipart request or response where the `message.body` variable is not changed, the result API action passes the attachments as the final response.
- If the `message.body` variable is changed, attachments are discarded.

- Retrieving an attachment
- Appending an attachment to the message
- Retrieving the number of attachments in an array
- Removing an attachment from the message
- Reading, accessing, and manipulating the body of an attachment
- Manipulating the headers of an attachment

Retrieving an attachment

GatewayScript examples

Retrieve the attachment by index location.

```
context.message.attachments[1]
```

Retrieve the attachment by looking up the value of `pic1` in the `Content-Id` header.

```
context.message.attachments['cid:pic1']
```

Retrieve the attachment by looking up the value of `uri1` in the `Content-Location` header.

```
context.message.attachments['uri1']
```

Appending an attachment to the message

GatewayScript example

```
var headers = JSON.parse ( '{ "Content-Type": "text/xml; charset=utf-8", "Content-Id": "<data>" }' );
context.message.attachments.append(headers, 'Hello world.');
```

Retrieving the number of attachments in an array

Example

```
for (var i = 0; i < context.message.attachments.count; i++) {
  context.message.attachments[i].header.set('X-Foo', 'bar');
}
```

Removing an attachment from the message

GatewayScript examples

Remove the attachment by index location.

```
context.message.attachments.remove(1)
```

Remove the attachment by looking up the value of `pic1` in the `Content-Id` header.

```
context.message.attachments.remove('cid:pic1')
```

Remove the attachment by looking up the value of `uri1` in the `Content-Location` header.

```
context.message.attachments.remove('uri1')
```

Reading, accessing, and manipulating the body of an attachment

GatewayScript examples

To read the payload of an attachment in binary and return the contents as either a `Buffer`, `Buffers`, `JSON`, or `XML NodeList` object:

```
context.message.attachments[0].body.readAsBuffer(function(errorObject,bufferObject){})
context.message.attachments[0].body.readAsBuffers(function(errorObject,buffersObject){})
context.message.attachments[0].body.readAsJSON(function(errorObject,jsonObject){})
context.message.attachments[0].body.readAsXML(function(errorObject,nodeList){})
```

To write content `Hello world` to the current payload of the attachment:

```
message.attachments[0].body.write("Hello world")
```

Manipulating the headers of an attachment

GatewayScript examples

To retrieve the values of the whole attachment headers:

```
context.message.attachments[0].headers
```

To retrieve the value of the named header of the attachment:

```
context.message.attachments[0].header.get();
```

To delete the current `Content-Type` header of the attachment.

```
context.message.attachments[0].header.remove('Content-Type');
```

You can set the value of the named header of an attachment. The header can be coalesced and noncoalesced. Take the following code as an example.

```
context.message.attachments[0].header.set('MyCookie', ['cookie1', 'cookie2']);
```

If the value type is array, the `MyCookie` header is noncoalesced and it is present with different entries in HTTP header.

```
MyCookie: cookie1
MyCookie: cookie2
```

User-defined policies

Make a user-defined policy available to a Catalog, and apply that policy to a REST or SOAP API.

A policy is a piece of configuration that controls a specific aspect of processing in the Gateway server during the handling of an API invocation at run time. IBM® API Connect provides built-in policies for many capabilities, including logging, redaction, proxy, and transformations, but you can also create user-defined policies to provide more processing control.

The following information outlines how to create and manage user-defined policies:

Creating a user-defined policy

A user-defined policy is created outside of API Connect by using a standard development tool, but the policy must conform to a particular schema. For information about how to create a policy, see Authoring policies.

Note: If you are running your policies on a DataPower® Gateway, you must ensure that the DataPower processing to be controlled by the user-defined policy, is supported by the level of IBM DataPower Gateway that the policy will be run on. When using the GatewayScript actions, the minimum DataPower firmware that should be used is version 7.2.0.

Importing a user-defined policy into API Connect

To make a user-defined policy available to an API, the policy must be imported into one or more Catalogs in the API Manager. If you also want to be able to apply the policy to your policy assembly in API Designer, you will have to import that policy into the developer toolkit environment.

Make your policy accessible to API Designer by using the developer toolkit. For detailed instructions, see Packaging and importing your policies into IBM API Connect.

Make your policy available by using the API Manager UI. For detailed instructions, see:

- Importing a user-defined policy into a Catalog. If you want to create a new Catalog and import a user-defined policy as part of the configuration, follow the instructions in Creating and configuring Catalogs.
- Packaging and importing your policies into IBM API Connect.

Note: You must import a user-defined policy into every Catalog in API Manager that you require the policy to run in.

Applying a user-defined policy to an API

Apply a user-defined policy to an API definition by using the assemble view in API Designer or API Manager, to add the component to your assembly and configure its properties. For detailed instructions, see Including elements in your OpenAPI 2.0 API assembly, Including elements in your OpenAPI 2.0 API assembly.

Deleting a user-defined policy

Delete a user-defined policy from a Catalog, by clicking the Delete icon next to the policy in the associated Catalog in API Manager.

Tags in API Connect

There are a number of forms of tagging in IBM® API Connect.

- You can create tags in an API that are visible in its OpenAPI definition and can be used to filter operations in the Developer Portal.
- You can tag items in the Developer Portal so that they can be searched for.

Tags within an API

The OpenAPI specification allows the inclusion of tags in an API definition, and you can create these tags through the API Manager user interface. You can assign tags to an operation within an API.

Tags that are assigned to an operation can be used to group the operations of an API in the Developer Portal, by using the filter options for the API. These tags are visible to anyone who can view any of the Products to which the API belongs.

For more information, see Editing an OpenAPI 2.0 API definition and Configuring an operation.

Note: Though you can tag the API itself in the API Setup section, by using this method the tag is not visible in the Developer Portal.

Tags in the Developer Portal

An administrator can assign tags to items in the Developer Portal, which are then available through the API Products search function in the Developer Portal.

For more information, see Editing tags for a specific item and Managing tags in the Developer Portal.

Related concepts

- Working with Products

Related tasks

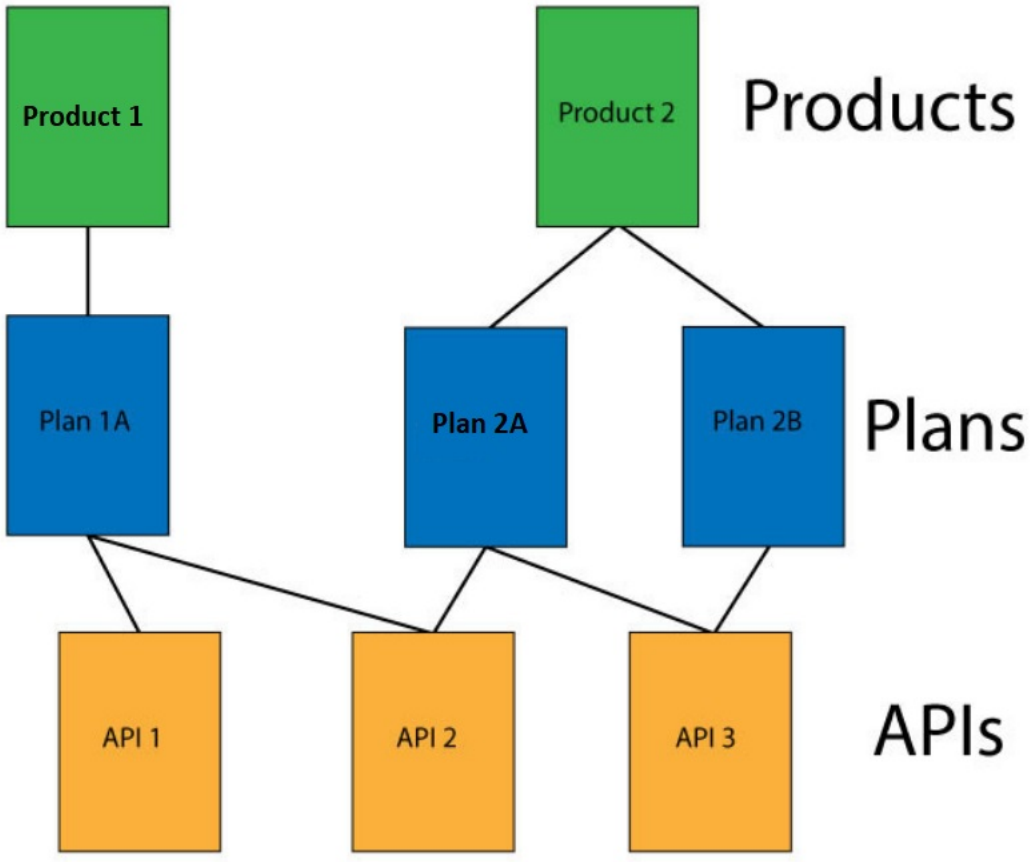
- Creating an API definition

Working with Products

In IBM® API Connect, Plans and APIs are grouped together in Products, with which you can manage the availability and visibility of APIs and Plans.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
The following diagram demonstrates how Products, Plans, and APIs relate to one another.
Note: Plans belong to only one Product, but they can possess different APIs to other Plans within the same Product, and they can share APIs with Plans from any Product.

Figure 1. The hierarchy of Products, Plans, and APIs.



Products provide a method by which you can group APIs into a package that is intended for a particular use. Additionally, they contain Plans, which can be used to differentiate between different offerings. Plans can share APIs, but whether subscription approval is required depends upon the Plan itself. Additionally, you can enforce rate limits through these Plans, or through operations within the APIs of a Plan that override the rate limit of the Plan. You can also assign different subscription costs to your Plans to differentiate the rates of API calls. For more information, see Monetizing your Products.

To make an API available to an application developer, it must be included in a Plan. You can create Plans only within Products, and these Products are then published in a Catalog. A lifecycle manager can then control the availability and visibility of APIs and Plans through the API Manager. By using the API Manager, you can specify the Plans that an application developer is able to subscribe to through the Developer Portal. The application developer can only subscribe to one Plan from a specific Product. Multiple Plans within a single Product are useful in that they can fulfill similar purposes but with differing levels of performance and cost. For example, you can have a "Demo Plan", which makes a single API available free of charge, and a "Full Plan" which makes several APIs available with a monthly subscription cost.

As well as controlling which APIs an application developer can use, different Plans can be used to implement different rate limits. A rate limit can be implemented as a default rate that is shared across an entire Plan, or can be set for specific operations of an API within that Plan, exempting them from the shared Plan rate limit. Different Plans can have differing rate limits, both between operations and for the overall limit. This is useful for providing differing levels of service to customers. For example, a "Demo Plan" might enforce a rate limit of ten calls per minute, while a "Full Plan" might permit up to 1000 calls per second.

In addition, you can apply burst limits to your Plans, to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals. You can also set multiple rate limits per Plan and per operation, at second, minute, hour, day, and week time intervals.

Note:

- Applying a rate limit at the Plan level creates a default rate limit that is shared across all the operations within the Plan. If you need to set specific rate limits for specific operations, you must set these within the operations themselves and these settings will override the setting at the Plan level.
- The test application that is used by the API Manager test tool is not subject to rate limits if you have enabled automatic subscriptions for the Catalog in which you are testing. For more information, see Working with Catalogs

API Connect also supports the implementation of multiple versions of Products. You can choose version numbers and use them to aid the development of your Products and Plans.
Note: The version for a Product is distinct from the version of any APIs that are contained in the associated Plans. Plans cannot themselves have their own version, they use the version of their parent Product.

- **Creating a draft Product**
  Create a draft Product to collect a set of APIs and Plans into one offering that you make available to your developers. A Plan includes rate limit settings that can be applied to the Plan as a whole, or specified for each operation in an API. Through Products and Plans, you have greater control over what APIs your developers have access to, and the subscription terms.
- **Downloading a draft Product**
  You can download the details of your draft Product so that you can store them for later recovery.
- **Importing a draft Product**
  You can create a Product by importing a Product definition in a .yaml file.
- **Searching for a draft Product**
  Quickly locate a draft API or Product by searching on the title and version.
- **Editing a draft Product**
  After initially creating a draft Product, you can continue to further configure the Product, or you can make configuration changes later.
- **Staging a draft Product**
  Stage a draft Product to a Catalog to create a specific version of that Product, before publishing. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers.. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.
- **Publishing a draft Product**
  Publish a draft Product to make the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.
- **Creating a new version of your Product**
  You can have multiple versions of a Product in IBM API Connect. These versions can occupy any of the lifecycle stages, which facilitates development.
- **Specifying the gateway type for a Product**
  You can edit the Product configuration to specify a gateway type.
- **Deleting a draft Product**
  You can delete draft Products that are no longer required.
- **Organizing your Products into categories**
  You can organize your Products into categories. The Products that you categorize are displayed within the Developer Portal, in their defined categories.

Related information

- Working with Products in the API Manager

Creating a draft Product

Create a draft Product to collect a set of APIs and Plans into one offering that you make available to your developers. A Plan includes rate limit settings that can be applied to the Plan as a whole, or specified for each operation in an API. Through Products and Plans, you have greater control over what APIs your developers have access to, and the subscription terms.

Before you begin

Define your APIs. For more information, see Creating an API definition.
Tip: As well as using the method described in this task, you can also create a Product when you create an API. If you create an API by using the developer toolkit command line editor, a Product is automatically created for you. You can then change any of the Product settings by opening your new Product in the Products page of the API Designer.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

When you first create your Product, it has *draft* status. When you have finished configuring your draft Product, you stage it to a Catalog so that the APIs in the Product can be made available to your application developers.

Procedure

To create a Product, complete the following steps:

1. In the navigation pane, click Develop, select the Products tab, then click Add > Product.
   The Add Product page opens.
2. Select New Product, then click Next.
3. On the Info page, complete the following steps:
   a. Enter the Title and Version of the Product and, optionally, a description. A Name is entered automatically.
      Note: The value in the Name field is a single string that is used to identify the Product in developer toolkit CLI commands. The Title is used for display.
      To view the CLI commands to manage draft Products, see apic draft-products.

   b. Click Next.
4. On the APIs page, select the APIs that you want to include in the Product. You can include only APIs whose gateway type matches the gateway type of the Product, or APIs for which the Enforced option is disabled, meaning that the API is not managed on an API Connect gateway. If you select an incompatible API whose gateway type does not match the gateway type of the Product, a warning message is displayed and you cannot save your changes until you clear the incompatible selection. For more information on gateway types, see API Connect gateway types, Specifying the gateway type for a Product, and Specifying the gateway type for an API definition.
5. Click Next.
6. On the Plans page, complete the following steps:
   a. Optional: Change the Title, Description, or Rate limit settings for the default Plan.
   b. Optional: To add further Plans, click Add, then scroll down and modify the Plan settings as required.
   c. Click Next.
7. To have the Product published to the Sandbox Catalog automatically after creation, select Publish product.
   Note: The publish option is not available in the API Designer UI.
8. If Publish product is selected, you can select which gateway service the Product is published to. By default, the Product is published to all relevant gateway services.
9. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.
   If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.

   Visibility can also be set at the Catalog or Space level. For more information, see Configuring Product visibility in a Catalog.

10. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.
    If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.
    Note: If you select custom visibility or subscribability, the consumer organizations and consumer organization groups that you can choose from are those in the Sandbox Catalog for the management server and provider organization that you are connected to. If you want to specify consumer organizations or groups that are in another Catalog, or in a Space, publish the Product to that Catalog or Space, then configure the visibility or subscribability there; for details, see Changing the availability of a Product.
    If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

    If you are working offline in API Designer, you must type in the exact name of the organization or group.

    For more information about consumer organizations in a Catalog, see Administering Consumer organizations.

11. Click Next to create your Product.
    a. To further configure your Product, click Edit Product.
       For details, see Editing a draft Product
    b. If you do not want to configure your Product further at this time, click the initial Develop link in the breadcrumb trail to return to the Develop page; you can then move on immediately to another task.
       For details on how to configure your Product later, see Editing a draft Product.

Results

You have created a draft Product, and specified a set of APIs and Plans into one offering that you can now make available to your developers.

What to do next

Stage your Product to a Catalog. For more information, see Staging a draft Product.

Related tasks

- Creating a new version of your Product

Related reference

- API and Product definition template examples

Related information

- Publishing a Product
- Working with Products in the API Manager
- Configuring Product visibility in a Catalog

Downloading a draft Product

You can download the details of your draft Product so that you can store them for later recovery.

Before you begin

To complete this task, you must have created a draft Product. For more information, see Creating a draft Product.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Product-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

You can download the details of your Product as a .yaml file. This file can be used to re-create your Product, although it includes only references to APIs, not the API definitions themselves. This function does not provide a way to export an entire implementation. For information about creating a Product from a .yaml file, see Importing a draft Product.

Procedure

To download a .yaml file containing details of your Product, complete the following instructions:

1. In the navigation pane, click Develop.
2. Select the Products tab.
3. Alongside the Product version that you want to work with, click the options icon ⋮ and then click Download.
4. Save the file to the required location.

Results

You have downloaded a .yaml representation of your Product.

Related tasks

- Importing a draft Product

Related information

- Creating a Product

Importing a draft Product

You can create a Product by importing a Product definition in a .yaml file.

Before you begin

To complete this task, you must have a .yaml representation of a Product. This can be created as described in the Downloading a draft Product task.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can import details of your Product as a .yaml file. This file can be used to re-create your Product, although it includes only references to APIs, not the API definitions themselves. This function does not provide a way to import an entire implementation.

You can import the file from your local file system.

Procedure

To create a Product from a .yaml file, complete the following instructions:

1. In the navigation pane, click Develop.
2. Select the Products tab.
3. Click Add > Product.
4. Select Existing Product, then click Next.
5. To import a file from your local file system, you can either drag and drop your files, or click Browse and select the file that you want to use.
   The wizard checks the validity of the YAML, and displays a message indicating successful validation.
6. Click Next.
7. To have the Product published to the Sandbox Catalog automatically after creation, select Publish product.
8. Click Next to create your Product.
   a. To further configure your Product, click Edit Product.
      For details, see Editing a draft Product
   b. If you do not want to configure your Product further at this time, click the initial Develop link in the breadcrumb trail to return to the Develop page; you can then move on immediately to another task.
      For details on how to configure your Product later, see Editing a draft Product.

Results

You have imported a Product from a .yaml file. If the same APIs and versions of APIs as referenced in the file are present, then they will be included in the Product.

Importing a Product in this way does not include the API definitions, only references to them.

Related tasks

- Downloading a draft Product

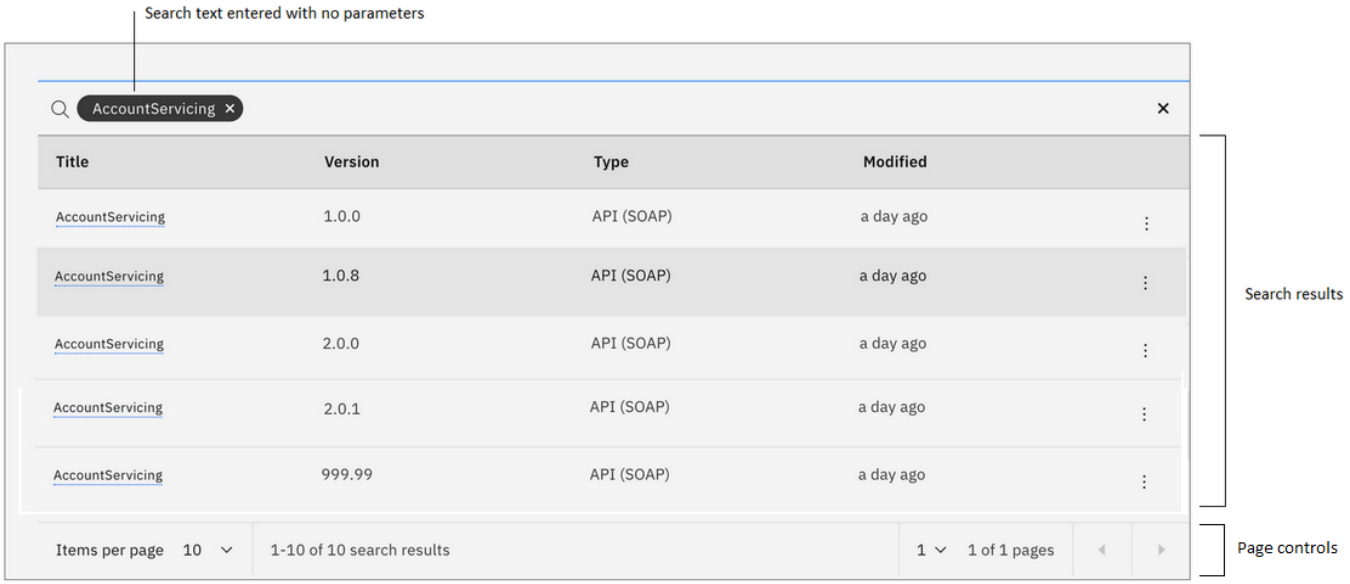
Searching for a draft Product

Quickly locate a draft API or Product by searching on the title and version.

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

On the Develop page, type your search string into the search field and click (or press Enter) to run the search.

The search returns a list of APIs (or Products) that most closely resemble your keywords, rather than limiting the results to identical matches. The following image shows a search string that includes "AccountServicing". The results include all draft APIs and Products with a title that is similar to the search text, with duplicates sorted by creation date.



Resources with a name, title, or version that are not similar to the search text are not included in the results. If you select a resource type to search for but do not specify any keywords, the results are sorted by title and then by creation date. For example, if you search for Products but do not include any keywords, then all Products appear in the results table.

Search text
: The search is case insensitive. Trailing spaces and accents (for example: é, è, â) are ignored. For best results, provide as much of the title and version as possible in your search text. Detailed search strings return more accurate results. Results include all titles that are similar to your search string, and not just identical matches.

What's included in the search?
: The scope of the search includes the API or Product Title, Name and Version fields. The search applies only to the current provider organization (displayed in the product banner's Organization field).

You can optionally restrict your search to Products only, or to a specific type of API. Click in the search field and then in the "Types" list that displays, select Product or an API type (REST or SOAP).

How are results displayed?

Search results display as a table, sorted by API or Product title and version. If the results overflow onto additional pages, navigate between pages using the standard page controls that follow the table. When you clear the search, APIs and Products are listed alphabetically by title and, within each title, by newest version first.

Editing a draft Product

After initially creating a draft Product, you can continue to further configure the Product, or you can make configuration changes later.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

During the initial creation of a Product, the Product creation wizard guides you to enter the minimum configuration settings, and then provides an Edit Product button. You have the option to specify additional configuration immediately, or to return to Product to edit it later. This topic describes how to specify additional configuration in either case.

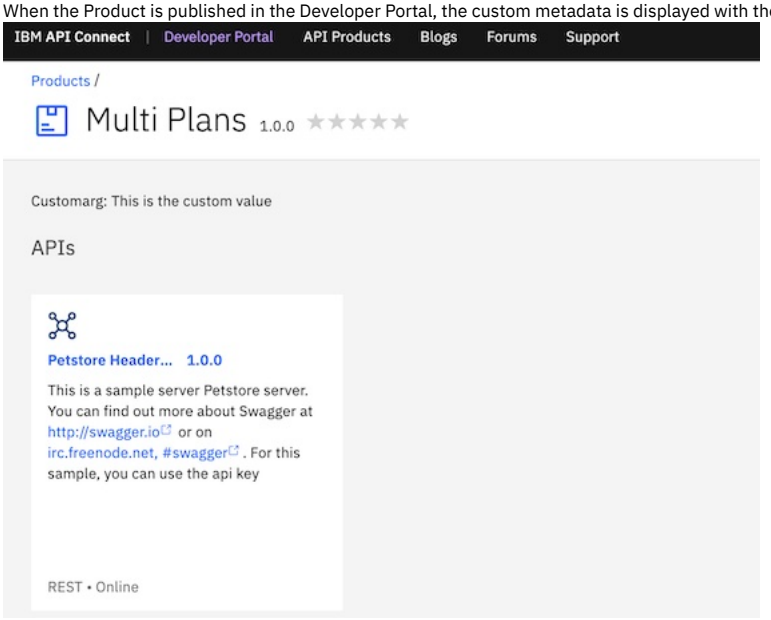
Procedure

To edit a draft Product, complete the following steps:

1. Access the Product Setup page in one of the following ways:
   - After completing the initial creation of the Product, click Edit Product.
   - In the navigation pane, click Develop, then select the Products tab.
   - Click the title of the Product that you want to edit.
2. On the Product Setup page, you can make the following configuration changes:
   a. In the Info section, change the Title, Version, or Summary.
   b. Use the Contact, Terms of Service, and License sections to enter the corresponding details as required.
   c. Change the gateway type. For more information, see API Connect gateway types and Specifying the gateway type for a Product.
   d. Click Save to save your changes.
3. Optional: Specify Product metadata to be displayed with the Product in the Developer Portal:
   a. Click the Source tab to see the OpenAPI source for the Product.
   b. Add the custom attribute `x-name: value` into the `info` section. For example:

```
info:
  version: 1.0.0
  title: Multi Plans
  name: multi-plans
  x-customarg: This is the custom value
...
```

   Where `customarg` is the name of the custom metadata, and `This is the custom value` is the information content.
   c. Click Save.
   When the Product is published in the Developer Portal, the custom metadata is displayed with the Product. For example:



4. Click Visibility, then make the following changes as required:
   a. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.
   If you select Custom, complete the following steps:

- Select the Catalog for which you want to control visibility.
- Use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.

b. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.
If you select Custom, complete the following steps:
- Select the Catalog for which you want to control subscribability.
- Use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be subscribable to. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.

Note:
- If you select custom visibility or subscribability, only 10 search results are displayed in the selection list for the consumer organizations and consumer organization groups. If necessary, enter a more specific search string to refine the search.
- Although the title of each organization or group is displayed, followed by the name in parentheses, in the table of selected organizations and groups only the name is displayed

a. Click Save to save your changes.

Note: You can change the visibility and subscribability settings when you publish the Product; see Publishing a draft Product. You can also change the settings in the Catalog to which the Product is published; see Changing the availability of a Product.

5. To specify the APIs that you want to include in the Product, click APIs, then complete the following steps:.
   a. Click Edit.
   All draft APIs are listed.
   b. Select the APIs that you want to include. You can include only APIs whose gateway type matches the gateway type of the Product, or APIs for which the Enforced option is disabled, meaning that the API is not managed on an API Connect gateway. If you select an incompatible API whose gateway type does not match the gateway type of the Product, a warning message is displayed and you cannot save your changes until you clear the incompatible selection. For more information on gateway types, see API Connect gateway types, Specifying the gateway type for a Product, and Specifying the gateway type for an API definition.
   c. Click Save when done.
   The selected APIs are listed.

   Note: To make an API available to application developers, you must include it in a Plan.
6. Optional: Add billing integration to the Product. Click Plans, and in the Billing integration section, select the billing integration resource that you want to apply to the Product.
   In the Plans section, you can then either edit the Default Plan to add pricing information, or create a new Plan with pricing information. For more information about billing integration, see Monetizing your Products.
7. Optional: Add one or more Plans to the Product, or modify an existing Plan. Note that a default plan is automatically created for you, with a rate limit of 100 API calls per hour.
   a. Click Plans.
   b. To add a new Plan, click Add. To modify an existing Plan, click the options icon alongside the required Plan, then click Edit.
   c. Specify the Title of the Plan and, optionally, a description. A Name is entered automatically.
   Note: The value in the Name field is a single string that is used to identify the Plan in developer toolkit CLI commands. The Title is used for display.
   d. Specify whether your Plan requires subscription approval. If you want subscriptions by developers to require approval, select the Approval check box; otherwise, ensure the check box is cleared.
   e. In the Plan pricing section, you can add pricing information by completing the following steps:
      i. Change the toggle to On for Plan pricing. The Plan pricing definition section is displayed.
      ii. If you want to include any free trial days in your Plan, select Include free trial days, and then enter the number of trial days that a subscriber can use the Plan without charge, after which their billing cycle begins.
      iii. Select the Currency for the billing process use.
      iv. Enter the Price per month to bill the subscriber for. If the selected currency supports fractional units, enter the price including any fractional units, such as cents.

   For more information about Product billing integration, see Defining a Product with billing integration.
   f. In the Plan Rate Limits section, you can modify a rate limit, and click Add to add further rate limits. You can set multiple rate limits, at second, minute, hour, day, and week time intervals. To remove a rate limit, click the corresponding delete icon .
   Note: For information about rate limits and burst limits in API Connect, see Understanding rate limits for APIs and Plans.
   g. In the Plan Burst Limits section, you can modify a burst limit, and click Add to add further burst limits. You use burst limits to prevent usage spikes that might damage infrastructure. You can set multiple burst limits, at second and minute intervals. To remove a burst limit, click the corresponding delete icon .
   Rate and burst limits work together to manage network traffic for the APIs covered under a Plan. A Plan can have multiple rate and burst limits, but it is recommended that each time interval be assigned only one set of limits. Adjust the rate and burst limits to allow for maximum traffic without overloading your network. The **rate limit** sets the maximum amount of sustainable, ongoing traffic for accessing the APIs on your network within a time interval (for example, one day). The **burst limit** sets the maximum short-term traffic volume for your network within a time interval (per second or minute).

   The **burst limit** allows for short bursts of increased traffic. When the **burst limit** is exceeded, all subsequent API calls are rejected until the start of the next **burst limit interval**. The **burst limit counter** is reset to zero at the start of the next interval, which allows API calls to be accepted again. These API calls count toward the **rate limit counter**, but the resetting of the **burst limit counter** does not affect the **rate limit counter**.

   The **rate limit** is the number of API calls allowed in a time interval, for example, 1000 calls per day. When the **rate limit** is exceeded, and Hard limit is enabled, all subsequent API calls will be rejected. The **rate limit counter** is reset to zero at the start of the next **rate limit interval**, which allows API calls to be accepted again. If Hard limit is disabled, all subsequent API calls are still accepted, and a message is logged stating that the **rate limit** has been exceeded. This is referred to as a "soft limit."

   Hard limit affects only the **rate limit**, as illustrated by the following scenarios:

   Scenario A

Table 1. Hard limit enabled

Hard limit	Burst limit	Rate limit
ON	100 calls/minute	1000 calls/day

- If a user calls an API 100 times in one minute, the **burst limit** is reached. The 101st call (and any subsequent calls) within the same minute will be rejected. Once the minute is up, the **burst limit counter** is reset. All API calls are tallied toward the **rate limit** of 1000 calls per day. The resetting of the **burst limit counter** does not affect the **rate limit counter**.
- If a user calls the API 99 times per minute, they will not hit the **burst limit**. They will eventually hit the 1000 calls per day rate limit, and the 1001st call will be rejected until the end of the same one day **rate limit interval**. During the period of time when API calls are rejected due to the daily **rate limit** being exceeded, the **burst limit** will not be activated since the calls are already rejected.
- Both **burst limits** and **rate limits** are applied per consumer.

Scenario B

Table 2. Hard limit disabled

Hard limit	Burst limit	Rate limit
OFF	100 calls/minute	1000 calls/day

- As in Scenario A, if a user calls the API 100 times in one minute, the 101st call within the same minute will be rejected, until that minute is up and the counter resets. These calls count toward the 1000 calls per day **rate limit** as well.
- If a user calls the API 99 times per minute, they will not hit the **burst limit**. They will eventually hit the 1000 calls per day **rate limit**, and the 1001st call will be accepted (since there is no hard limit). A message will be logged for each subsequent call until the time interval (one day) is up and the counter resets. During the remainder of the day, the **burst limit** will still be enforced, and calls will be rejected once the number of calls exceeds the 100 calls per minute within any given minute.
- Both burst limits and rate limits are applied per consumer.
  Note: When Hard limit is unchecked, the **rate limit** is considered a "soft limit." With a soft limit, calls are not rejected after the **rate limit** is reached. Instead, a message is recorded in the log file. With a soft limit, the **burst limit** still rejects API calls after it is exceeded.

h. To include in the Plan all the APIs that were included in the Product in step 5, select Same as product in the Plan APIs section.
i. In the GraphQL Rate Limits section you can configure rate limits that will be applied to any GraphQL proxy APIs that are added to the Product.
The following rate limits are available:

graphql-field-cost
: Applies a limit to the total calculated field cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

graphql-design-request
: Applies a limit to the total number of calls of any of the following types made to the GraphQL proxy APIs in this Plan:
  - Requests sent to the `graphql/cost` endpoint for retrieving the cost of a GraphQL query. For more information, see Enabling the cost endpoint for a GraphQL API.
  - Standard introspection requests sent to the `/graphql` endpoint. For more information, see Supporting introspection for a GraphQL API.
  - HTML web browser requests for a GraphiQL editor sent to the `/graphql` endpoint. For more information, see Enabling the GraphiQL editor for a GraphQL API.

graphql-input-type-cost
: Applies a limit to the total calculated input type cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

graphql-type-cost
: Applies a limit to the total calculated type cost of GraphQL query calls made to the GraphQL proxy APIs in this Plan.

The calculated costs depend on weighting factors applied to the types and fields in the GraphQL schema. For details on the GraphQL proxy API, and configuring type and field weights, see Creating a GraphQL proxy API and Using the GraphQL schema editor.

j. To define one or more assembly burst limits or assembly count limits, click Add in the Assembly Burst Limits or Assembly Count Limits section as appropriate. For a burst limit you specify the maximum number of calls allowed in a specified time period, for a count limit you specify a maximum call limit. The name that you supply is for use in a Rate Limit policy in an API assembly, enabling the policy to define the burst or count limit that is to be applied; for more information, see Rate Limit.
k. To select the APIs that you want to include in the Plan, select Customize the plan API list in the Plan APIs section, then click Edit. The Edit plan APIs window opens. The APIs that are available for selection are those that were included in the Product in step 5. Use the check boxes to specify the required APIs, then click Save when done.
l. To select specific API operations to include in the Plan, click the options icon ⋮ alongside the required API , then click Edit operation list. The Select operations you want to include window opens. Use the check boxes to specify the required operations, then click Save when done.
m. To remove an API from the Plan, click the options icon ⋮ alongside the required API in the Plan APIs section, then click Remove.
n. For any GraphQL APIs in the Plan, if you want to prevent subscribers to this Plan from using certain types or fields, complete the following steps:
   i. Click the options icon ⋮ alongside the required GraphQL API in the Plan APIs section, then click Edit show/hide settings.
   ii. To prevent use of a GraphQL type, clear the check box selection alongside that type.
   iii. To prevent use a of a field in a type, expand the type then clear the check box selection alongside that field.
   iv. When done, click Next. A summary window opens. If the operation is permitted, any related elements that will also be hidden are listed; for example, if you hide a type, and that type is referenced as the data type of a field on another type, that field is also hidden. If the operation is not permitted, an explanation is displayed.
   v. If the operation is permitted, click Done to apply your changes. If the operation is not permitted, either click Cancel to close the window or click Back and amend your settings.

   You can use the Edit show/hide settings option to change existing settings; you can hide additional types or fields, and you can show types or fields that were previously hidden.

o. To define rate limits for specific API operations, select Override plan rate limits for individual operation in the Plan APIs section; for further details, see Defining rate limits for an API operation
p. Click Save to save your changes.

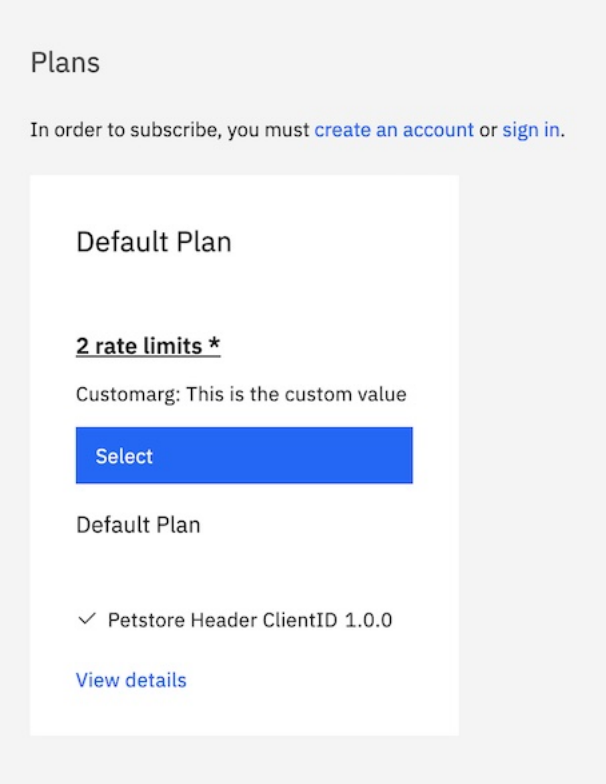
8. Optional: Specify Plan metadata to be displayed with the Plan in the Developer Portal:
   a. Click the Source tab to see the OpenAPI source for the Plan.
   b. Add the custom attribute `x-name: value` into the appropriate `plan` section. For example:

```
...
plans:
  default-plan:
    title: Default Plan
    description: Default Plan
    x-customarg: This is the custom value
    rate-limits:
```

```
      default:
        value: 100/1hour
        hard-limit: false
      minute:
        value: 20/1minute
        hard-limit: false
    apis:
      petstore-header-clientid1.0.0: {}
...
```

Where ***customarg*** is the name of the custom metadata, and ***This is the custom value*** is the information content.

c. Click Save.
When the Plan is published in the Developer Portal, the custom metadata is displayed with the Plan. For example:



9. Optional: Click Categories, then define any categories that you want to organize your Product into. Click Save to save any updates.
By organizing your Products into categories, you can provide a hierarchical display for your Products in the Developer Portal. For more information, see Organizing your Products into categories.

What to do next

Stage your Product to a Catalog. For more information, see Staging a draft Product.

- **Defining rate limits for an API operation**
For any Plan in a draft Product, you can apply one or more rate limits to a specific operation on any of the APIs in that Plan.

Related tasks

- Creating a new version of your Product

Related reference

- API and Product definition template examples

Related information

- Publishing a Product
- Working with Products in the API Manager

Defining rate limits for an API operation

For any Plan in a draft Product, you can apply one or more rate limits to a specific operation on any of the APIs in that Plan.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

During the initial creation of a Product, the Product creation wizard guides you to enter the minimum configuration settings, and then provides an Edit Product button. You have the option to specify additional configuration immediately, or to return to Product to edit it later. This topic describes how to specify additional configuration in either case.

Note: For information about rate limits and burst limits in API Connect, see Understanding rate limits for APIs and Plans.

Procedure

1. Access the Product Setup page in one of the following ways:
   - After completing the initial creation of the Product, click Edit Product.
   - In the navigation pane, click Develop, then select the Products tab.
   - Click the title of the Product that you want to edit.
2. Click Plans.
3. Click the options icon ⋮ alongside the required Plan, then click Edit.
4. Scroll down to the Plan APIs section, then select Override plan rate limits for individual operation.
   The Override Rate Limits section is displayed.
5. Click Add. The Override plan rate limits window opens.
6. Select the API, and the specific operation in that API, to which you want to apply rate limits.
   By default, calls to the operation are unlimited.
7. To apply one or more rate limits to the operation, select Rate Limits.
8. To define a rate limit, click Add.
9. Supply a name for the rate limit, and define the maximum number of calls allowed in a specified time period; for example, 100 calls per 1 minute. If you select HARD LIMIT and the limit is exceeded, subsequent calls are rejected, otherwise calls are still accepted and a message is logged.
10. Click Save when done.
    An entry for the operation is listed in the Override Rate Limits section.
11. To modify the rate limiting definition for an operation, click the options icon ⋮ alongside the required operation, then click Edit Operation; to delete it, click Remove.
12. Click Save to save your changes.

Staging a draft Product

Stage a draft Product to a Catalog to create a specific version of that Product, before publishing. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers.. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be staged only to a Space within that Catalog.

Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see Creating and configuring Catalogs.
Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in IBM API Connect®.
To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Stage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see Creating and configuring Catalogs or Managing user access in a Space.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.
Staging is not available when working offline in API Designer.

A Catalog is a staging target, and behaves as a logical partition of the DataPower® Gateway, and the Developer Portal.

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests. For information on publishing a Product, see Publishing a Product.

You cannot re-stage an existing Product version to a production Catalog. If you update an API, you must create a new version of that API. Then, create a new version of the Product so that it includes the newer version of the API. For more information on creating new versions of APIs and Products, see Creating a new version of an API definition and Creating a new version of your Product. If Spaces are enabled in a production Catalog, you cannot re-stage the same API or Product version to any Space in the Catalog.

You can, however, re-stage a Product version to the Sandbox Catalog, or other non-production Catalog, for ease of iterative testing. Note that when you re-stage a Product version to a non-production Catalog, any application subscriptions are deleted; this is done to facilitate scripted deployment in a continuous integration environment, where the re-creation of application subscriptions will be controlled by the scripts. If Spaces are enabled in a non-production Catalog and you re-stage a Product version to a different Space to the one in which it was previously staged, it is removed from the previous Space and staged to the newly specified Space. If billing integration is enabled in a Product, you cannot re-stage the same Product version into a non-production, or production, Catalog. For more information about billing integration, see Monetizing your Products.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (https://www.npmjs.com/package/@apidevtools/swagger-parser).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either **`X-IBM-Client-Id`** or **`X-IBM-Client-Secret`**.

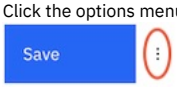
- Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
- De-referencing of local references in the definition file (that is, values of `$ref` properties), and ensuring these are valid JSON Pointers within the file.

Note: If the OpenAPI file that defines your API uses a `$ref` field to reference a fragment of OpenAPI code that is defined in a separate file, the `$ref` field is replaced with the contents of the target file before the product that contains the API is staged or published (the `$ref` field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Procedure

To stage a Product, complete the following steps:

1. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
2. In the navigation pane, click Develop.
3. Select the Products tab.
4. You can stage a Product either from the Develop listing page, or from within the Product itself.
   a. To stage a Product from the Develop listing page, click the options menu icon alongside the required Product version, and then select Stage.
   b. To stage a Product from within the Product itself, complete the following steps:
      i. Click the Product version that you want to work with.
      ii. Click the options menu icon:

      iii. Select Stage.
5. Select the Catalog to which you want to stage the Product.
6. If Spaces have been enabled in the selected Catalog, select the Space that you require.
   Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to.
   If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

   For details of how to create a Catalog in a provider organization, see Creating and configuring Catalogs.
7. If, when the staged Product is subsequently published, you want it to be published only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see API Connect gateway types.
8. Click Stage.

Results

Your Product is staged to a Catalog. You can view the state of the Product in the Catalog in API Manager. If you staged the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Staged.

For information about the lifecycle of a product, see The Product lifecycle.

If approval is required to stage Products in the Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is staged when the request is approved. If approval is not required, the Product is staged immediately.

For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

What to do next

- Publish your Product for application developers to access it in the Developer Portal. For more information, see Publishing a Product.

Related information

- Managing your Products
- Removing a Product from a Catalog
- Working with Products in the API Manager

Publishing a draft Product

Publish a draft Product to make the APIs in that Product visible on the Developer Portal for use by application developers. The syndication feature in IBM® API Connect means that if Spaces are enabled for a Catalog, Products can be published only to a Space within that Catalog.

Before you begin

Ensure that you have a Catalog to stage to in the API Manager or API Designer user interfaces (UI). For more information, see Creating and configuring Catalogs.
Note: All references in this topic to a Catalog can also be applied to Spaces in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in IBM API Connect®.

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Product > Manage permission for the target Catalog or Space. For information on configuring Product management permissions for a Catalog or Space, see Creating and configuring Catalogs or Managing user access in a Space.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI. Publishing is not available when working offline in API Designer.

You cannot re-publish a Product version to a production Catalog, you must create a new version of the Product for publication; see Creating a new version of your Product. If Spaces are enabled in a production Catalog, you cannot re-publish the same Product version to any Space in the Catalog, neither the same Space nor a different Space.

You can, however, re-publish a Product version to the Sandbox Catalog, or other non-production Catalog, for ease of iterative testing. When you re-publish a Product version to a non-production Catalog, you can choose whether or not to preserve any application subscriptions; deleting subscriptions can facilitate scripted deployment in a continuous integration environment, where the re-creation of application subscriptions will be controlled by the scripts.

If Spaces are enabled in a non-production Catalog and you re-publish a Product version to a different Space to the one in which it was previously published, it is removed from the previous Space and published to the newly specified Space.

Validation of the API definition file occurs during the staging or publishing process. The following validation occurs:

- Validation against the OpenAPI schema by using the API Dev Tools Swagger Parser (https://www.npmjs.com/package/@apidevtools/swagger-parser).
- Validation against IBM extension properties.
- Semantic validation, which includes the following types of validation:
  - Ensuring that if an OpenAPI is enforced by an API Connect Gateway, then the scheme must be HTTPS, or the parameter name for an API key security scheme in the header must be either **`X-IBM-Client-Id`** or **`X-IBM-Client-Secret`**.
  - Ensuring that if the OpenAPI is not enforced by an API Connect Gateway, then a "host" must be provided.
  - De-referencing of local references in the definition file (that is, values of **`$ref`** properties), and ensuring these are valid JSON Pointers within the file.

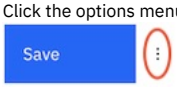
Note: If the OpenAPI file that defines your API uses a **`$ref`** field to reference a fragment of OpenAPI code that is defined in a separate file, the **`$ref`** field is replaced with the contents of the target file before the product that contains the API is staged or published (the **`$ref`** field is supported only if you are using the API Connect for IBM Cloud developer toolkit). For more information, see Using $ref to reuse code fragments in your OpenAPI files.

Procedure

To publish a Product, complete the following steps:

1. **Optional:** If you have accounts on multiple provider organizations, you can select a new provider organization for staging and publishing from the Organization menu.
2. In the navigation pane, click Develop.
3. Select the Products tab.
4. You can publish a Product either from the Develop listing page, or from within the Product itself.
   a. To publish a Product from the Develop listing page, click the options menu icon ⋮ alongside the required Product version, and then select Publish.
   If you are re-publishing a Product to non-production Catalog, and the Product has application subscriptions, the Publish option deletes them. To preserve application subscriptions, click Publish (Preserve Subscriptions) instead.
   Note: If there are any existing subscriptions to the Product, the Publish (Preserve Subscriptions) operation will fail if you remove any consumer organizations or consumer organization groups from the Product visibility or subscribability settings; you can, however, add further organizations or groups.
   For information on managing visibility and subscribability settings, see Editing a draft Product.

   For information on consumer organization groups, see Working with consumer organization groups.

   b. To publish a Product from within the Product itself, complete the following steps:
      i. Click the Product version that you want to work with.
      ii. Click the options menu icon:
      

      iii. Select Publish.
5. Select the Catalog to which you want to publish the Product. You must have a compatible gateway type associated with the catalog to proceed, as described in Creating and configuring Catalogs.
6. If Spaces have been enabled in the selected Catalog, select the Space that you require.
   Note: The Catalogs that you can select from are those that are defined for the management server and provider organization that you are connected to.
   If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

   For details of how to create a Catalog in a provider organization, see Creating and configuring Catalogs.

7. If you want to publish the Product only to selected gateway services, select Publish to specific gateway services, then select the required gateway services. Only the gateway services whose type matches the gateway type setting for the Product are listed. For information on gateway types, see API Connect gateway types.
8. The Visibility and Subscribability sections display the visibility and subscribability settings that have been configured for the draft Product, but you can modify these before publishing if required; proceed as follows:
   a. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.
   If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.
   b. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.

If you select Custom, use the Search organizations and groups field to search for the consumer organizations and consumer organization groups, in the selected Catalog, that you want the Product to be subscribable to. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see Administering Consumer organizations.
Note:

- If you select custom visibility or subscribability, only 10 search results are displayed in the selection list for the consumer organizations and consumer organization groups. If necessary, enter a more specific search string to refine the search.
- If custom visibility or subscribability has already been configured in the draft Product, the titles that are displayed in the table of selected organizations and groups are derived by taking the names that are defined in the draft Product and finding the corresponding titles in the Catalog that has been selected as the publish target.

9. Click Publish.

Results

Your Product is published to a Catalog. The state of the Product is shown as Published in the header of the Product itself, and you can click Published to see which Catalogs or Spaces the Product is published to.

You can also view the state of the Product in the Catalog in API Manager. If you published the product from API Designer, ensure you are logged into API Manager with the same user name and password that you used for API Designer. Click Manage in the API Manager UI, then select the required Catalog. The Product is shown with a state of Published.

For information about the lifecycle of a product, see The Product lifecycle.

If approval is required to publish Products in the Catalog, a publish approval request is sent, and the Product moves to the Pending state; the Product is published in the Catalog when the request is approved. If approval is not required, the Product is published immediately.

Note: If approval is required to stage Products to the Catalog, a stage approval request is sent. When the request is approved, the Product moves to the staged state and must be separately published in the Catalog. For information on publishing a staged Product in a Catalog, see Publishing a Product.
For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.

If the Product contains no Plans, a Plan called Default Plan is added automatically to the Product in the Catalog.

Related information

- Managing your Products
- Removing a Product from a Catalog
- Working with Products in the API Manager

Creating a new version of your Product

You can have multiple versions of a Product in IBM® API Connect. These versions can occupy any of the lifecycle stages, which facilitates development.

Before you begin

Create your Product and assign it a version. For more information, see Creating a draft Product.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can create multiple versions of a Product, and these versions can be named according to your own preferences, and function as distinct Products when staged.

Procedure

To create a new version of a Product, complete the following steps:

1. In the navigation pane, click Develop.
2. Select the Products tab.
3. Alongside the Product version that you want to work with, click the options icon, then click Save as a new version.
   The Save Product as a new version window opens.
4. Enter your new version number, then click Save.

Results

Your Product is saved as a new version.

Related information

- Managing your Products
- Working with Products in the API Manager

Specifying the gateway type for a Product

You can edit the Product configuration to specify a gateway type.

Before you begin

IBM® API Connect supports two gateway types: DataPower® Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see API Connect gateway types.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

You can specify the gateway type for a Product.

Products can use only one type of gateway, and the APIs in the Product must use the same type of gateway. See Specifying the gateway type for an API definition.

When you create a new Product, or import a Product definition that has no gateway type configured, the gateway type defaults according to the gateway types of the gateway services that are enabled in your Sandbox Catalog, as indicated in the following table.

Table 1. Default gateway type for a new Product

Gateway Service enablement in Sandbox Catalog	Default gateway type for new Product
No gateway services enabled	DataPower API Gateway
At least one gateway service of each type enabled	DataPower API Gateway
One or more gateway services only of type DataPower API Gateway enabled	DataPower API Gateway
One or more gateway services only of type DataPower Gateway (v5 compatible) enabled	DataPower Gateway (v5 compatible)

If you import a Product definition that already has a gateway type configured, that gateway type is retained.
For details of how to enable gateway services in a Catalog, see Creating and configuring Catalogs.

Procedure

To specify the gateway type for a Product, complete the following steps:

1. Ensure that you know the type of gateway that your APIs are configured to use.
2. In the navigation pane, click Develop, then select the Products tab.
3. Click the title of the Product you want to update.
4. On the Product Setup page, go to the Gateway Type section. Select the gateway type for your Product:
   - DataPower Gateway (v5 compatible)
   - DataPower API Gateway

   Note:
   You can only select the gateway type that satisfies the selected enforced APIs in your Product. To change the APIs included in your product, see Editing a draft Product.
5. Click Save to retain your changes.

Deleting a draft Product

You can delete draft Products that are no longer required.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

API Manager UI only: To complete this task, you must be assigned a role that has the **`Product-Drafts:Edit`** permission. The pre-supplied Developer role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Note:

- Deleting a draft Product does not remove it from any Catalogs to which it has been published. For details on how to remove a Product from a Catalog, see Removing a Product from a Catalog.
- You cannot directly delete a Product that was generated automatically from an API with the Activate API option; these Products have the title *api_title* auto product. To delete such Products, delete the API from which it was generated; the Product is deleted together with the API. For more information, see Deleting an API definition.

Procedure

1. In the navigation pane, click Develop.
2. Select the Products tab.

3. Alongside the Product version that you want to delete, click the options icon ⋮ and then click Delete.

Organizing your Products into categories

You can organize your Products into categories. The Products that you categorize are displayed within the Developer Portal, in their defined categories.

Before you begin

Your role must have the necessary permissions to stage and publish Products.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

By organizing your Products into categories, you can provide a hierarchical display for your Products in the Developer Portal.
Note: Organizing Products into a hierarchical view in the API Designer or API Manager UI is different to tagging in the Developer Portal.

Procedure

1. In the navigation pane, click Develop, then select the Products tab..
2. Click the title of the Product that you want to work with.
3. Click Categories.
4. In the Categories text box, enter the hierarchical taxonomy path that you want your Product to follow, in the following format:

   top_level_element / next_level_element / x_level_element

   For example:

   `Animals / Fluffy / Cat`

5. After you have completed entering the category specifications for the Product, click Save.

What to do next

- Publish the Product; see Publishing a draft Product.
- After publishing the Product, you can, in the Developer Portal, display Products in the categories that you have defined. For more information, see Displaying APIs and Products in categories.

Creating and validating API and Product definitions by using the CLI

The developer toolkit of IBM® API Connect provides a command line interface that you can use to create and publish API and Product definitions, and also to validate YAML or JSON definitions.

The following topics explain how to use the command line interface to create an OpenAPI definition file, create a Product definition file, and validate a YAML or JSON definition.

- **Creating an OpenAPI definition file**
  APIs are defined in OpenAPI definition files, in YAML format. You can create a default OpenAPI definition file by using the **create** command and then modify it by using an editor of your choice.
- **Specifying a gateway type for an API definition**
  An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). API definitions must specify which type of gateway the API uses.
- **Referring to an extension in an API definition**
  To refer to an OpenAPI extension in your API definition YAML (or JSON) file, add an **`extensions`** key under **`x-ibm-configuration`**.
- **Using $ref to reuse code fragments in your OpenAPI files**
  If you deploy an API to an IBM API Connect Management server by using the developer toolkit command line, you can use the **`$ref`** field in your OpenAPI YAML and JSON API definition files to reference a fragment of OpenAPI code that is defined in a separate file. When API Connect processes the source API definition file, the **`$ref`** field is replaced with the contents of the target file.
- **Creating a Product definition file**
  You define a Product by creating a Product definition file.
- **Using x-ibm-languages to create multilingual API and Product documentation**
  Create multilingual API and Product documentation by using the **`x-ibm-languages`** extension in your API and Product OpenAPI definitions.
- **Using x-example to control the examples displayed in the Developer Portal**
  OpenAPI does not support the use of **`example`** attributes on parameters, only on request and response objects and their properties. To allow API Developers to be able to control the examples displayed in the portal you can use **`x-example`** in OpenAPI parameters.
- **Using x-embedded-doc to add additional documentation to products and APIs**
  You can use **`x-embedded-doc`** to add additional documentation to a product or an API as part of their definition.
- **Using x-pathalias to give consistent URLs for products and APIs**
  You can use **`x-pathalias`** to assign a URL to a product or an API.
- **Validating the YAML or JSON definition of an API or Product**
  You can validate a YAML or JSON definition by using the IBM API Connect developer toolkit.
- **Creating and using API and Product definitions templates**
  You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates

containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

Related reference

- API development and management commands

Creating an OpenAPI definition file

APIs are defined in OpenAPI definition files, in YAML format. You can create a default OpenAPI definition file by using the **create** command and then modify it by using an editor of your choice.

You can stage or publish the API directly to a Catalog in API Manager by referencing the API in a Product definition file, and then using the **`apic products:publish`** command to publish the Product. You can also create a draft API in API Manager by using the **apic draft-apis:create** command.

You can create an API in the CLI by running **`apic create:api`**, and supplying additional arguments on the command line.

Another option is to create an API interactively in the command line by running **`apic create:api`** and following the prompts.

You can see further details and available options for the **`apic create:api`** command by running:

```
apic create:api --help
```

IBM provides an extension to the OpenAPI specification; this extension is described in IBM extensions to the OpenAPI specification.

Note: Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "`(?`" cannot be validated or published. An error message is returned. For example:

```
Product has not been published!
The multipart 'openapi' field contains an OpenAPI definition with validation errors.
    definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0, col:
0)
400
```

Creating an API definition from a template

You can use a custom Handlebars template to create an API by using the following command:

```
apic create:api --template template_filename --title api_title
```

where *template_filename* is the name of the Handlebars template to use, and *api_title* is the title of your API.
An API template file must have a .hbs file name extension. You can create a template from scratch, or start with the example (default) API template provided in API and Product definition template examples.

You can create multilingual API and Product documentation by using an **`x-ibm-languages`** extension directly in the OpenAPI definition. For more information, see Using x-ibm-languages to create multilingual API and Product documentation.

- **IBM extensions to the OpenAPI specification**
  You use the **`x-ibm-configuration`** object in your OpenAPI definition file to add extensions that are specific to IBM API Connect.

Related reference

- API development and management commands

IBM extensions to the OpenAPI specification

You use the **`x-ibm-configuration`** object in your OpenAPI definition file to add extensions that are specific to IBM® API Connect.

The x-ibm-configuration extension has the following structure:

```
x-ibm-configuration:
  enforced: enforced_boolean
  phase: Phase
  testable: test_boolean
  cors:
    enabled: cors_boolean
  activity-log:
      activity_logging_extension
  assembly:
    execute:
      - assembly_component
  gateway: gateway_type
  type: api_type
  properties:
      properties_extension
  catalogs:
      catalogs_extension
```

The following table lists the various extensions used by API Connect, whether they are required, a description of their use and behavior, and their type.

Table 1. IBM extensions

Extension	Required	Description	Type
phase	No	Use the phase extension to describe the maturity of the API. It can take the following values: • **identified**: the API is in the early conceptual phase and is neither fully designed nor implemented. • **specified**: the API has been fully designed and passed an internal milestone but has not yet been implemented. • **realized**: the API is in the implementation phase.	String
testable	No	Use the **testable** extension to specify whether the API can be tested using the test tool in the Developer Portal.	Boolean
enforced	No	Use the **enforced** extension to specify if the API Connect gateway is used to enforce the API. • **true** indicates that the API Connect gateway is used to enforce the API. • **false** indicates that the API Connect gateway is not used to enforce the API.	Boolean
cors	No	Use the **cors** extension to specify whether CORS access control is used for the API. The extension has an **enabled** field which is a Boolean.	Object (with a single Boolean field)
API Gateway only activity-log	No	Use the **activity-log** extension to configure your logging preferences for the API activity that is stored in analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.	Object
assembly	No	Use the **assembly** extension to describe the application of policies and logic to an API. It contains an **execute** field that contains an array of policies that are applied in order. It can contain a **catch** field that contains an array of error cases to be caught. For information about use of the **execute** field, see execute. For information about use of the **catch** field, see catch.	Object
gateway	No	Use the **gateway** extension to specify which type of gateway you want to use. If you are using the DataPower® Gateway (v5 compatible) or DataPower API Gateway, it must be included and take one of the following values: • `datapower-gateway` (DataPower Gateway (v5 compatible)) • `datapower-api-gateway` (DataPower API Gateway) For information about types of gateways, see API Connect gateway types.	String
type	No	The **type** extension takes one of the following values: • **rest** • **wsdl**	
properties	No	Use the **properties** extension to define properties for use in an API.	Object (properties)
catalogs	No	Use the **catalogs** extension to define Catalog-specific values for properties defined in the **properties** extension.	Object (catalogs)

The following example shows the **x-ibm-extension** section of an API that is enforced by API Connect, is in the realized state, is testable through the test tool in the Developer Portal, has CORS access control enabled, and has a simple assembly that invokes a URL and then redacts a field from the request or response.

```
x-ibm-configuration:
  enforced: true
  phase: realized
  testable: true
  cors:
    enabled: true
  assembly:
    execute:
      - invoke:
          title: Example Invoke
          target-url: 'https://example.com/api'
          description: Example description
      - redact:
          actions:
            - action: redact
              from:
                - request
                - response
              path: //*[@name='secondaryAddress']/*[@name='streetAddress']
  properties:
    ID:
      value: 1234
      description: An ID to be used for validating.
      encoded: false
  catalogs:
    Sandbox:
      properties:
        ID: 5678
```

- **catch**
  Use the catch extension to catch errors that occur during an API call.
- **properties**
  Use the **properties** extension to define properties for referencing in an API.
- **catalogs**
  Use the **catalogs** extension to assign catalog-specific values to properties defined in the **properties** section.
- **activity-log**
  If you're using the DataPower API Gateway, you can use the **activity-log** extension to configure your logging preferences for the API activity that is stored in analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

execute

The execute field of an assembly has the following structure:

```
execute:
 - Policy_1
 - Policy_2
```

Note: Although some built-in policies can be used with both the DataPower® Gateway (v5 compatible) and the DataPower API Gateway, some policies are restricted to a particular Gateway. The following icons indicate which Gateway each policy can be used with:

- DataPower Gateway Indicates that the policy can be run on the DataPower Gateway (v5 compatible).
- API Gateway Indicates that the policy can be run on the DataPower API Gateway.

For details of the two types of gateway, see API Connect gateway types.

The following table describes the possible policies and logic constructs that can be included in an execute field.

Table 1. execute properties

Property	Required	Description	Data Type	DataPower Gateway	API Gateway
activity-log	No	Use the activity-log policy to log information that relates to the calling of API operations.	object (activity-log)	✓	✓ Functionality provided by the activity-log extension
client-security	No	Provides a range of options for authenticating client access to your APIs, extending the capabilities of the OpenAPI specification.	object (client-security)	✗	✓
gatewayscript	No	Include a GatewayScript program.	object (gatewayscript)	✓	✓
graphql-introspect	No	Use the graphql-introspect policy to introspect a GraphQL schema.	object (graphql-introspect)	✗	✓
if	No	Use the if policy to execute a section of the assembly only when a condition is fulfilled.	object (if)	✓	✓ Functionality provided by switch
invoke	No	Use the invoke policy to call an API. The last invoke in your policy might be replaced by a proxy automatically to improve performance. To disable this, see: API properties.	object (invoke)	✓	✓
json-to-xml	No	Convert payload from JSON to XML.	object (json-to-xml)	✓	✓
jwt-generate	No	Generate a JSON Web Token (JWT).	object (jwt-generate)	✓	✓
jwt-validate	No	Validate a JSON Web Token (JWT).	object (jwt-validate)	✓	✓
log	No	Use the log policy to customize or override the default activity logging configuration for an API.	object (log)	✗	✓
map	No	Use the map policy to transform variables.	object (map)	✓	✓
operation-switch	No	Use the operation-switch policy when you want to execute alternative policy assemblies, conditional on the operation that is being called.	object (operation-switch)	✓	✓
oauth	No	Use the oauth policy to policy to perform OAuth processing based on defined OAuth provider settings.	object (oauth)	✗	✓
parse	No	Use the parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.	object (parse)	✗	✓
proxy	No	Proxy a service.	object (proxy)	✓	✓ Functionality provided by invoke
ratelimit	No	Use the ratelimit policy to apply one or more rate or burst limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls that an application can make to an API in a specified time period.	object (ratelimit)	✗	✓
redact	No	Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.	object (redact - DataPower API Gatewayredact - DataPower Gateway (v5 compatible))	✓	✓
set-variable	No	Use the set-variable policy to set a runtime variable to a string value, or to add or clear a runtime variable.	object (set-variable)	✓	✓
switch	No	Use the switch policy to execute one of a number of sections of the assembly based on which specified condition is fulfilled.	object (switch)	✓	✓
throw	No	Use the throw policy to specify points at which an error should be thrown.	object (throw)	✓	✓
user-defined-policy	No	You can apply your own user-defined policies to your APIs.	object (User-defined policies)	✓	✓
user-security	No	Extract a user's credentials, authenticate those credentials, and obtain authorization from the user.	object (user-security)	✗	✓

Property	Required	Description	Data Type	DataPower Gateway	API Gateway
validate	No	Use the Validate policy to validate the payload in an assembly flow against a JSON or an XML schema.	object (validate - DataPower API Gateway, validate - DataPower Gateway (v5 compatible))	✓	✓
validate-usernametoken	No	Validate a WS-Security UsernameToken.	object (validate-usernametoken)	✓	✗
xml-to-json	No	Convert payload from XML to JSON.	object (xml-to-json)	✓	✓
xslt	No	Apply an XSLT transform to the payload.	object (xslt)	✓	✓

The following example shows an execute field for an assembly that invokes a URL and then redacts a field from the request or response.

```
execute:
 - invoke:
  title: Example Invoke
  target-url: 'https://example.com/api'
  description: Example description
 - redact:
  actions:
   - action: redact
     from:
      - request
      - response
    path: //*[@name='secondaryAddress']/*[@name='streetAddress']
```

activity-log

Use the Activity Log policy to configure your logging preferences for the API activity that is stored in IBM® API Connect analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Activity Log.

Note: If you are using the DataPower API Gateway, you configure your logging preferences by using the activity-log extension.

About

The activity-log policy has the following format:

```
- activity-log:
  version: version
  title: title
  description: description
  content: activity_to_log_if_call_successful
  error-content: activity_to_log_if_call_unsuccessful
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

You can also apply an activity-log policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see Activity Log in the built-in policies section.

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
  success-content: activity_to_log_if_call_successful
  error-content: activity_to_log_if_call_unsuccessful
  enabled: is_activity_logging_enabled
buffering: true
```

Properties

The following table describes the policy properties:

Table 2. Activity Log policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	A title for the policy.	string
description	No	A description of the policy.	string

Property	Required	Description	Data type
content	Yes	Defines the type of content to be logged when the operation is successful. Valid values: • **none**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **activity**: Logs invocation only (only the resource URI is recorded). • **header**: Logs activity and header. • **payload**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **activity**.	string
error-content	No	Indicates what content to log if an error occurs. Valid values: • **none**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **activity**: Logs invocation only (only the resource URI is recorded). • **header**: Logs activity and header. • **payload**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **payload**.	string

Example 1

```
# use defaults

- activity-log:
  version: 1.0.0
  title: default activity logging
```

Example 2

```
- activity-log:
  version: 1.0.0
  title: no logging for successful calls
  content: none
  error-content: activity
```

client-security

The client-security policy can be used to extend client authentication access for your APIs.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Client Security.

About

You use the client-security policy to create an assembly action that's separate from the OpenAPI specification, to allow for more options to authenticate an application. Note that you cannot use the client-security policy to perform rate-limiting enforcement.

In a client security action, you define the following settings.

- Control whether to stop processing if client security fails. When client security fails, stops assembly processing and returns an error.
- Control whether to require the client secret. When required, the secret is compared to the registered secret on the application that is identified by the client ID.
- Define the method to extract client credentials from the request.
  - For all methods except **http**, use the id-name and the secret-name properties to specify the location that contains the ID and the location that contains the secret.
    - When **cookie**, specify which cookie.
    - When **context-var**, specify which runtime context variable.
    - When **form**, specify the form data.
    - When **header**, specify which header.
    - When **query**, specify which query parameter.
  - For the **http** method, use the http-type property to specify the format of the authorization header, which expects the **basic** form in the **basic base64_id:secret** format.
- Define the method to authenticate the extracted client credentials. The supported methods are **native** (this means by using IBM® API Connect), or by using a specified **third-party** user registry.
  - If **third-party**, use the user-registry property to specify the user-registry to authenticate the extracted client credentials. The supported registry types are LDAP and authentication URL.

Properties

The client-security policy has the following format:

```
- client-security:
    version: version
    title: title
    description: description
    stop-on-error: is_processing_stopped_on_client_security_failure
    secret-required: is_client_secret_required_in_request
    extract-credential-method: method_for_supplying_credentials
    id-name: parameter_that_specifies_client_id
    secret-name: parameter_that_specifies_client_secret
    http-type: authentication_type
    client-auth-method: method_for_client_authentication
    user-registry: user_registry_for_client_authentication
```

Table 2. client-security policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
stop-on-error	Yes	If set to **true**, assembly processing stops if client security fails, and an error is returned.	boolean
secret-required	Yes	If set to **true**, the client secret must be sent in the request. The secret is compared to the registered secret on the application that is identified by the client ID.	boolean
extract-credential-method	Yes	Specify one of the following values to define how the calling application authenticates: • **header**: client ID and client secret credentials must be supplied in the request header. • **query**: client ID and client secret credentials must be supplied as query parameters in the request URL. • **form**: client ID and client secret credentials must be supplied as form data sent in a POST request. • **cookie**: client ID and client secret credentials must be supplied in a header called **Cookie**. • **http**: the calling application must authenticate by using basic authentication. • **context-var**: the credentials that are used for authenticating the client are obtained from context variables that you set in the assembly flow prior to the Client Security policy, by using a gatewayscript policy for example. The names of these context variables are determined by the values that you supply in the id-name and secret-name properties of the client-security policy.	string
id-name	Yes, unless extract-credential-method is set to **http**	The name of the parameter whose value specifies the client ID. For all values of extract-credential-method other than **context-var** and **http**, the calling application must supply a parameter with this name, in the location defined by the extract-credential-method setting. For **context-var**, this property specifies the name of a context variable. This option does not apply if the extract-credential-method property is set to **http**.	string
secret-name	Yes, if secret-required is set to **true** and extract-credential-method is other than **http**	The name of the parameter whose value specifies the client secret. For all values of extract-credential-method other than **context-var** and **http**, the calling application must supply a parameter with this name, in the location defined by the extract-credential-method setting. For **context-var**, this property specifies the name of a context variable. This option does not apply if the extract-credential-method property is set to **http**.	string
http-type	Yes, if extract-credential-method is set to **http**	The authentication type. Currently, this must be set to **basic**.	string
client-auth-method	Yes	Specify one of the following values: • **native**: only client ID and client secret are used to authenticate the request. If extract-credential-method is set to **http** then the calling application must supply the client ID for the user name, and the client secret for the password. • **third-party**: a user registry is used to authenticate the client. If extract-credential-method is set to a value other than **http** then the calling application must supply the user name for the client ID, and the password for the client secret.	string
user-registry	Yes, if the client-auth-method is set to **third-party**	Specify the value of the **name** property of the user registry that will be used to authenticate the client. The supported registry types are LDAP and authentication URL.	string

client-security policy example

```
- client-security:
    version: 2.0.0
    title: client-security
    stop-on-error: true
    secret-required: true
    extract-credential-method: cookie
    id-name: my-client-id
    secret-name: my-client-secret
    client-auth-method: third-party
    user-registry: myauthurl
```

extract

Use the extract policy to extract and transform data from fields in the API context.

Gateway support

Table 1. Gateways that support this policy and the corresponding policy versions

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Extract.

About

The extract policy specifies the data source that contains the content to transform, the fields that contain the content, and expressions that define how to transform the content. You use a subset of JSONata notation to specify the fields to extract and transform. For more information, see Constructing JSONata expressions to extract and transform data.

The input to the extract policy must be parsed data. One way to produce parsed data is to add a parse policy before the extract policy in your assembly.

The extract policy has the following format:

```
- extract:
    version: 2.0.0
    title: title
    root: data source
    extracts:
      - capture: path expression that identifies the field
        transform: expression that defines how to transform the content
                   .
                   .
                   .
    description: description
```

Apply this policy by adding an `assembly` extension with an `execute` field to your OpenAPI definition file.

Properties

Table 2. extract policy properties

Property label	Required	Description	Data type
version	Yes	The policy version number. You must specify a version for the policy that is compatible with the gateway that you are using. When the API is published, if the version is incompatible with the gateway, a validation error is thrown that specifies the available versions.	string
title	Yes	The title of the policy. The default value is `extract`.	string
description	No	A description of the policy.	string
root	Yes	The data source that contains the content to transform. The default value is `message.body`.	string
capture	Yes	The path expression that identifies the field. The default setting is `$`, which indicates the entire input.	string
transform	Yes	The expression that defines how to transform the content.	string

Example

Transform the contents of the `account` field to include only the last 4 characters.

```
- extract:
    version: 2.0.0
    title: extract
    root: message.body
    extracts:
      - capture: $.members.policy.**.account
        transform: $substring($,-4)
    description: Include only the last 4 characters of the account field.
```

gatewayscript

Use the gatewayscript policy to execute a specified DataPower GatewayScript program.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see GatewayScript.

About

The gatewayscript policy has the following structure:

```
- gatewayscript:
    version: version
    title: Title
    desription: Description
    source: Script
```

Properties

The following table describes the policy properties:

Table 2. gatewayscript policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
source	Yes	The GatewayScript source code to execute.	string

Example

The following is an example of a simple gatewayscript policy:

```
- gatewayscript:
    version: 1.0.0
    title: Example_GatewayScript
    source: console.debug('Hello World!');
    description: A simple GatewayScript policy.
```

For more information about how to use a gatewayscript policy, see GatewayScript in the built-in policies section.

For more code examples, see GatewayScript code examples and, if you are using the DataPower API Gateway, Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

For general information on using GatewayScript, see the following topics in the DataPower product documentation:

- GatewayScript APIs for API management
- Accessing and manipulating a variable in the API context
- OAuth context variables

Related concepts

- Variable references in API Connect

Related reference

- GatewayScript code examples
- API Connect context variables

graphql-introspect

Use the graphql-introspect policy to introspect a GraphQL schema.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see GraphQL introspect.

Properties

Table 2. graphql-introspect policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
input	No	A variable in the API context that contains the input to introspect. The content of the **body** field of the variable is the input to introspect. By default, the variable name is **message**.	string
output	No	A variable in the API context where the results of the introspection are stored. The content of the **body** field of the variable is the result of GraphQL introspection. The default variable name is the same as that of the variable specified for the **input** field. Therefore, by default, the input of the introspection is overwritten by the output.	string

Example

```
- graphql-introspect:
    version: 2.0.0
    title: introspect the accounts graphql schema
    input: message
```

if

Use the if construct when you want to execute a portion of your assembly only when a specific condition is fulfilled.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by switch	

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see if.

About

The if policy has the following format:

```
- if:
  version: version
  title: title
  description: description
  condition: 'condition_1
      execute:
        policy_assembly_1 ...
```

In the condition field, use the form **apim.getvariable('*context.location.variable*')** to reference your variables, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

The **execute:** section can define any policy assembly, including further if policies. For more information, see execute.

Properties

Table 2. if policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
condition	Yes	A script that returns **true** or **false**. Use GatewayScript for a DataPower Gateway implementation. For information about the context variables you can use and how to reference them in your script, see API Connect context variables.	string
execute	Yes	The policy assembly that you want to execute if the condition returns **true**. For more information, see execute.	string

Example

```
# carry out different redaction actions depending on the operation

- if:
  version: 1.0.0
  title: clear_region_and_set_body
  condition: 'apim.getvariable('request.body.secret') == true'
    execute:
      - redact:
          title: remove secret field
          actions:
            - action: remove
              from: all
              path: /document/user/secret
```

invoke

Use the invoke policy to call an API.

Gateway support

Restriction: It is not possible to cache a request that includes the Authorization request header.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.1 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Invoke.

About

The invoke policy does not support responses with multipart form data, that is, when the response is set to `Content-Type: multipart/related`.

The invoke policy has the following format:

```
- invoke:
  version: version
  title: title
  description: description
  target-url: URL_of_target_API
  backend-type: how_payload_is_sent_to_backend
  tls-profile: TLS_profile_to_be_used
  verb: method_type
  timeout: timeout_value_in_seconds
  compression: is_data_to_be_compressed
  username: username_if_authentication_required
  password: password_if_authentication_required
  output: location_of_the_invoke_result
  cache-key: unique_identifier_of_the_document_cache_entry
  cache-response: cache_behavior
  cache-putpost-response: response_caching_behavior
  cache-ttl: cache_time_to_live
  inject-proxy-headers: are_proxy_headers_sent_to_target_url
  decode-request-params: are_request_parameters_decoded
  encode-plus-char: are_plus_characters_encoded
  keep-payload: is_payload_sent_on_delete
  use-http-10: are_transactions_restricted_to_http_1.0
  chunked-uploads: are_chunked_encoded_documents_sent_to_the_server
  persistent-connection: are_persistent_connections_enabled
  header-control:
        .
        .
        .
    headers_to_copy_to_target_url
        .
        .
        .
  parameter-control:
        .
        .
        .
    parameters_to_copy_to_target_url
        .
        .
        .
  follow-redirects: url_redirection_behavior
  stop-on-error: errors_that_stop_the_flow
```

Properties

The following table describes the properties of the invoke policy.

Table 2. Invoke policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
target-url	Yes	The URL of the target API.	string

Property	Required	Description	Data type
API Gateway only backend-type	No	Specify one of the following values to determine how the payload is sent to the backend: • **detect**: detect the message type and send the payload to the backend accordingly. • **xml**: package the payload as XML and send. If the message type is not XML, the operation fails. • **json**: package the payload as JSON and send. If the message type is not JSON, the operation fails. • **binary**: package the payload as binary and send, regardless of the message type. • **graphql**: package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails. The default value is **detect**.	string
API Gateway only graphql-send-type(policy version 2.2.0 and later)	Yes, if backend-type is set to **graphql** or **detect**, and verb is set to **POST** or **keep**.	Specify one of the following values to determine how the GraphQL payload is sent to the back end: • **detect**: detect the message type and send the payload to the back end accordingly. • **graphql**: package the payload as GraphQL and send. If the message type is not GraphQL, the operation fails. • **json**: package the payload as JSON and send. If the message type is not GraphQL, the operation fails. Note: graphql-send-type is supported only if backend-type is set to **graphql** or **detect**, and verb is set to **POST** or **keep**.	string
tls-profile	No	Specifies a TLS profile to use for the secure transmission of data. Note: If you want to edit the API in the source view, make sure that the tls-profile value follows this format: name:version. For instance, it should look like nameofmytlsclientprofile:1.0.0.	string
verb	No	The operation method type. Valid values: • GET • POST • PUT • DELETE • PATCH • HEAD • OPTIONS The default value is **GET**. However, if the property is omitted from the source, the HTTP method from the incoming request is used.	string
timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is **60**.	integer
API Gateway only http-version (policy version 2.1.0 and later)	Yes	The HTTP version that will be used when connecting to the backend server. Valid values: • **HTTP/1.0** • **HTTP/1.1** • **HTTP/2**	string
API Gateway only http2-required (policy version 2.1.0 and later)	No	Set this property to **true** to require that the server selects HTTP/2 during a TLS connection, otherwise the DataPower API Gateway rejects the connection and fails the request. For a non TLS connection, a warning is logged stating that the requirement can't be enforced. This setting applies only if http-version is set to **HTTP/2**.	boolean
compression	No	Specifies whether data is to be compressed by using compress before it is uploaded. The default value is **false**.	boolean
username	No	The username to use for HTTP Basic authentication.	string
password	No	The password to use for HTTP Basic authentication.	string
output	No	The name of a variable that will be used to store the response data from the request. By default, the invoke response, that is the body, headers, statusCode and statusMessage, is saved in the variable *message*. Use this property to specify an alternate location to store the invoke response. This variable can then be referenced in other actions, such as map. Note: If you want the response to be saved in *message*, leave the output property blank, do **not** supply the value `message`.	string
cache-key	No	Specifies the unique identifier of the document cache entry.	string
API Gateway only websocket-upgrade (policy version 2.1.0 and later)	No	Set this property to **true** to allow an HTTP or HTTPS connection to be upgraded to the WebSocket protocol if the request received by the corresponding HTTP or HTTPS handler on the DataPower API Gateway uses WebSocket (ws) or WebSocket Secure (wss). Note: • This property is available only for GraphQL APIs. • The **websocket-upgrade** property is deprecated. To allow WebSocket upgrade requests that can manage API processing data, define an assembly WebSocket upgrade policy. For more information, see websocket-upgrade.	boolean

Property	Required	Description	Data type
cache-response	No	The cache response type. Valid values: • **protocol**: The cache behavior is defined by the Cache-Control headers on the request and response. • **no-cache**: Specifies that there is no caching. However, if the document is already in the cache, the document is retrieved from the cache. • **time-to-live**: Specifies that the response stays in the cache for the specified time. The default value is **protocol**.	string
cache-putpost-response	No	Specifies whether to cache the response from POST and PUT requests. Caching the response from POST and PUT requests can reduce server load and reduce latency in the response to the client request. The default value is **false**.	boolean
cache-ttl	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property cache-response is set to **time-to-live**. Enter a value in the range 5 - 31708800. The default value is **900**.	integer
API Gateway only inject-proxy-headers	No	When set to true, the **invoke** policy injects the **X-Forwarded-For**, **X-Forwarded-To**, **X-Forwarded-Host**, and **X-Forwarded-Proto** headers to the request that is sent to the t**target-url**. The default value is **false**.	boolean
API Gateway only decode-request-params	No	When set to true, any request parameters that are referenced by a variable definition on the **target-url** of the invoke policy are URL-decoded. The default value is **false**.	boolean
API Gateway only encode-plus-char	No	When set to true, all "+" characters in the query parameter values of the **target-url** are encoded to %2F. The default value is **false**.	boolean
API Gateway only keep-payload	No	When set to true, the invoke policy sends a payload on an **HTTP DELETE** method. The default value is **false**.	boolean
API Gateway only use-http-10 (policy version 2.0.0 only)	No	When set to true, HTTP transactions are restricted to version 1.0. The default value is **false**.	boolean
API Gateway only chunked-uploads	No	If you set this property to **true**, chunked-encoded documents are sent to the server. When the HTTP 1.1 protocol is used, the body of the document can be delimited by either **Content-Length** or chunked encoding. While all servers can interpret **Content-Length**, many applications fail to understand chunked-encoded documents. For this reason, **Content-Length** is the standard method. The use of **Content-Length** interferes with the ability of the DataPower Gateway to fully stream. If you must stream full documents to the target server, enable this property. When enabled, the server must be RFC 2616 compatible. Unlike all other HTTP 1.1 features that can be negotiated down at run time, you must know beforehand that the target server is RFC 2616 compatible. The default value is **true**. Note: Chunked encoding is not supported by the HTTP 1.0 protocol.	boolean
API Gateway only persistent-connection (policy version 2.3.0 and later)	No	Set this property to **true** to enable HTTP persistent connections, allowing connection re-use. The default value is **true**.	boolean
API Gateway only **header-control:** **type** **values**	No	Specifies the headers in **message.headers** that you want to copy to the target URL. If the **type** property is set to **blocklist**, the **values** property lists the headers that you don't want to be copied. If the **values** property is empty then all headers are copied. If the **type** property is set to **allowlist**, the **values** property lists the headers that you want to be copied. The items that are listed in the **values** property are in regular expression format. The values are not case-sensitive. The default value of the **header-control** property is **header-control:** **type: blocklist** **values: []** See header-control examples	object
API Gateway only **parameter-control:** **type** **values**	No	Specifies the parameters in the incoming request that you want to be copied to the target URL. If the **type** property is set to **blocklist**, the **values** property lists the parameters that you don't want to be copied. If the **type** property is set to **allowlist**, the **values** property lists the parameters that you want to be copied. If the **values** property is empty then no parameters are copied. The items that are listed in the **values** property are in regular expression format. The values are not case-sensitive. The default value of the **parameter-control** property is **parameter-control:** **type: allowlist** **values: []** See parameter-control examples	object

Property	Required	Description	Data type
API Gateway only follow-redirect	No	Specifies the behavior if the back-end server returns the HTTP status code `301 Moved Permanently`. If this property is set to `true`, the `invoke` policy follows the URL redirection by making a further call to the URL specified in the `Location` header in the response. If this property is set to `false`, the `invoke` saves the `301` status code and the API call is considered to be complete. Note: The `follow-redirect` property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the `invoke` always follows the URL redirection; the `proxy` policy (not supported by the DataPower API Gateway) saves the `301` status code and completes the API call without following the URL redirection.	boolean
stop-on-error	No	List the errors that, if thrown during the policy execution, cause the flow to stop. If there is a `catch` flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors specified for the Stop on error property, or if the error thrown is not one of the specified errors, the policy execution is allowed to complete, and the assembly flow continues.	string

Example

```
- invoke:
    version: 2.0.0
    title: get the account status
    target-url: https://example.com/accounts/{id}?status={status}
    cache-response: time-to-live
    cache-putpost-response: true
    tls-profile: MyTLSProfile:1.0.0
    verb: POST
    timeout: 60
    compression: false
    username: MyUser
    password: MyPassword
    stop-on-error:
      - ConnectionError
      - OperationError
```

API Gateway only `header-control` examples

```
# copy all headers to the target URL

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: blocklist
      values: []

# copy all headers except X-Client-ID and Content-Type

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: blocklist
      values:
        - ^X-Client-ID$
        - ^Content-Type$

# copy no headers

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: allowlist
      values: []

# copy only the Content-Type header

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: allowlist
      values:
        - ^Content-Type$
```

API Gateway only `parameter-control` examples

```
# copy no request parameters to the target URL

- invoke:
    target-url: http://myhost/path?storeid=3
    parameter-control:
      type: allowlist
      values: []

# append the petid parameter to the target URL
# if the incoming request is http://apigw/org/sandbox/petstore/base?petid=100&display=detailed,
# the target URL at runtime will be http://myhost/mypath?storeid=3&petid=100

- invoke:
    target-url: http://myhost/path?storeid=3
    parameter-control:
      type: allowlist
```

```
      values:
        - ^petid$
```

json-to-xml

Use the json-to-xml policy to convert the context payload of your API from the JavaScript Object Notation (JSON) format to the extensible markup language (XML) format.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see JSON to XML.

About

The json-to-xml policy has the following structure:

```
- json-to-xml:
   version: version
   title: Title
   description: Description
```

Note: If you are using the DataPower API Gateway, the input to the json-to-xml policy must be parsed data. One way to produce parsed data is to use a **parse** policy before a json-to-xml policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table describes the policy properties:

Table 2. Policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
API Gateway only input	No	The input message to convert. Specify the name of a variable in the API context. ***variableName*.body**, the message payload, represents the JSON input to convert. The default value of the variable is **message** and **message.body** is the default input.	string
API Gateway only output	No	The output message to store the conversion result. Specify the name of a variable in the API context. ***variableName*.body** represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is **message.body**. Otherwise, when the input message is the variable **my-message-variable**, for example, the default output is **my-message-variable.body**. The variable cannot be any read-only in the API context.	string
API Gateway only conversionType	No	The conversion type that determines the target format of the output. The following options are available: • None: No conversion of the output takes place. • badgerFish: BadgerFish convention is used to determine the target conversion format of the output.	string
root-element-name	Yes	The root element name of the resultant XML document. This property is used only if the input JSON document is not hierarchical and has more than one uppermost level property, or if the always-output-root-element property is set to **true**.	
always-output-root-element	Yes	Select this property to **true** you always want the policy to output the root element, even if it is not required to make the XML document well formed.	boolean
unnamed-element-name	No	The XML element name to be used for JSON array elements.	string

Example

The following is an example of a json-to-xml policy:

```
- json-to-xml:
   version: 1.0.0
   title: JSON to XML transform
   description: Transforms JSON message body to XML format
```

jwt-generate

Use the Generate JWT security policy in IBM® API Connect to generate a JSON Web Token (JWT).

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Generate JWT.

Note:

- For algorithm types HS256, HS384, and HS512 the cryptographic objects referenced must be a Shared Secret Key.
- For algorithm types RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512 the cryptographic objects referenced must be a Crypto Key (private key).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to sign the JWT.

About

The jwt-generate policy has the following structure:

```
- jwt-generate:
  version: version
  title: title
  description: description
  jwt: json_web_token
  jti-claim: jwt_id_claim
  iss-claim: issuer_claim
  exp-claim: validity_period
  sub-claim: subject_claim
  aud-claim: audience_claim
  jws-jwk: sign_jwk_variable_name
  jws-alg: cryptographic_algorithm
  jws-crypto: sign_crypto_object
  jwe-enc: encryption_algorithm
  jwe-jwk: encrypt_jwk_variable_name
  jwe-alg: key_encryption_algorithm
  jwe-crypto: encrypt_crypto_object
```

Properties

The following table describes the policy properties:

Table 2. Generate JWT policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
jwt	No	Runtime variable in which to place the JWT that is generated. The default value is: **`generated.jwt`**. However, if not set, the JWT that is generated is written to the Authorization Header as a Bearer token.	string
jti-claim	No	Indicates whether a JWT ID (jti) claim should be added to the JWT. If selected, the property is set to **`true`**, and a UUID is generated and set as the JTI claim value.	boolean
iss-claim	Yes	Runtime variable from which the Issuer (iss) claim string can be retrieved. This claim represents the Principal that issued the JWT. The default value is: **`iss.claim`**	string
sub-claim	No	Runtime variable from which the Subject (sub) claim string can be retrieved.	string
aud-claim	No	Runtime variable from which the Audience (aud) claim string can be retrieved. Multiple variables are set by using a comma-separated string.	string
exp-claim	Yes	The length of time (in seconds), that is added to the current date and time, in which the JWT is considered valid. The default value is **`3600`**.	integer
private-claims	No	Runtime variable from which a valid set of JSON claims can be retrieved. These claims are added to any set of claims specified previously.	string
jws-jwk	No	Runtime variable that contains the JWK that is used to sign the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-alg	No	The cryptographic algorithm to use. Valid values are: • HS256 • HS384 • HS512 • RS256 • RS384 • RS512 • ES256 • ES384 • ES512 • PS256 • PS384 • PS512	string

Property	Required	Description	Data type
jws-crypto	No	The cryptographic object to use to sign the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to sign the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jwe-enc	No	The encryption algorithm to use. Valid values are: • A128CBC-HS256 • A192CBC-HS384 • A256CBC-HS512	string
jwe-jwk	No	Runtime variable that contains the JWK to use to encrypt the JWT.	string
jwe-alg	No	The key encryption algorithm to use. Valid values are: • RSA1_5 • RSA-OAEP • RSA-OAEP-256 • dir • A128KW • A192KW • A256KW	string
jwe-crypto	No	The cryptographic object to use to encrypt the claim.	string

Example

The following is an example of a jwt-generate policy:

```
- jwt-generate:
    version: 1.0.0
    title: jwt-generate
    iss-claim: iss.claim
    exp-claim: 3600
    jwt: generated.jwt
    jti-claim: true
    sub-claim: sub.claim
    aud-claim: aud.claim
    private-claims: private.claims
    jws-jwk: jws.jwk
    jws-alg: HS256
    jws-crypto: jwsCryptoObjectName
    jwe-enc: A128CBC-HS256
    jwe-jwk: jwe.jwk
    jwe-alg: A128KW
    jwe-crypto: jweCryptoObjectName
```

For more information about how to use a jwt-generate security policy, see Generate JWT in the built-in policies section.

jwt-validate

Use the Validate JWT security policy to enable the validation of a JSON Web Token (JWT) in a request before allowing access to the APIs.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Validate JWT.

Note:

- If the original message was signed with a Shared Secret Key, the cryptographic object that is specified must also be a Shared Secret Key.
- If the original message was signed with a Private Key, the cryptographic object that is specified must be a Crypto Certificate (public certificate).
- The cryptographic material can be provided through a JSON Web Key (JWK).
- If a JWK header parameter is included in the header of the JWT, the parameter must match the JWK or cryptographic object that is specified in the policy, or the JWT validation will fail.
- If both a cryptographic object and a JWK are specified, the cryptographic object is used to decrypt or verify the JWT.
- The JWT validate action on the DataPower API Gateway can verify a JWT by using either a single JWK, or a JWK set.

About

The jwt-validate policy has the following structure:

```
- jwt-validate:
  version: version
  title: title
  description: description
  jwt: json_web_token
```

```
output-claims: output_full_set_of_jwt_claims
iss-claim: issuer_claim
aud-claim: audience_claim
jwe-crypto: decrypt_crypto_object
jwe-jwk: decrypt_crypto_jwk_variable_name
jws-crypto: verify_crypto_object
jws-jwk: verify_crypto_jwk_variable_name
```

Properties

The following table describes the policy properties:

Table 2. Validate JWT policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
jwt	Yes	Context or runtime variable that contains the JWT to be validated. The default value is: **`request.headers.authorization`**. However, if this property is not set, the policy looks for the JWT in the **`request.headers.authorization`** location by default. Note: The format of the authorization header must be: **`"Authorization: Bearer jwt-token"`** where *jwt-token* is the encoded JWT.	string
output-claims	Yes	Runtime variable to which the full set of claims that are contained in the JWT is assigned. The default value is: **`decoded.claims`**.	string
iss-claim	No	The Perl Compatible Regular Expressions (PCRE) to use to validate the Issuer (iss) claim.	string
aud-claim	No	The PCRE to use to validate the Audience (aud) claim.	string
jwe-crypto	No	The cryptographic object (a shared key or certificate) to use to decode the claim. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jwe-jwk	No	Runtime variable that contains the JWK to use to decrypt the JWT. A JWK and a Crypto Object are both valid ways of providing the cryptographic data necessary to decrypt the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-crypto	No	The cryptographic object (a shared key or certificate) to use to verify the signature. A JWK, or a JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.	string
jws-jwk	No	Runtime variable that contains the JWK, or a JWK set, to use to verify the signature. A JWK, or a JWK set, and a Crypto Object are both valid ways of providing the cryptographic data necessary to verify the JWT. However, if both data types are specified, only the Crypto Object is used.	string

Example

The following is an example of a jwt-validate policy:

```
- jwt-validate:
    version: 1.0.0
    title: jwt-validate
    jwt: request.headers.authorization
    output-claims: decoded.claims
    iss-claim: "'^data.*'"
    aud-claim: "'^id.*'"
    jwe-crypto: jweCryptoObjectName
    jwe-jwk: jwe.jwk
    jws-crypto: jwsCryptoObjectName
    jws-jwk: jws.jwk
```

For more information about how to use a jwt-validate security policy, see Validate JWT in the built-in policies section.

API Gateway only

log

Use the log policy to customize or override the default activity logging configuration for an API.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Log.

About

The log policy has the following format:

```
- log:
  version: version
  title: title
  description: description
  mode: activity_logging_actions
  log-level: type_of_content_to_log
```

Properties

The following table describes the properties of the log policy.

Table 2. log policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string

mode	Yes	Specify one of the following values:
		• **gather-only**: gather all analytics data and write it to the **log** context variable, which populates the API event record on completion of the API execution. For more information on the fields in the **log** context variable, and the consequent API event record, see API event record field reference.
		• **send-only**: perform the following actions:
		  ◦ Read the data from the **log** context variable.
		  ◦ Truncate all message payloads and convert to a textual representation.
		  ◦ Send the data to the analytics server.
		• **gather-and-send**; perform a **gather-only** operation, immediately followed by a **send-only** operation.
		If you use the Send-only or Gather-and-send option, data is buffered and sent to the analytics server in batches according to the time interval configured for the Analytics Endpoint on the DataPower API Gateway. For more information, see Configuring an analytics endpoint in the DataPower knowledge center.
		Note: If you are offloading to a third party analytics server, you can redact any aspect of the event data. If you are using API Connect analytics, you can redact only request and response payloads.
			string

log-level	No	The type of content to log. Specify one of the following values: • **none**: Indicates no logging. • **activity**; Logs the invocation only, which is only the resource URI. • **header**: Logs activity and header. • **payload**:Logs activity, header, and payload. • **default**; Use the log level setting defined in the API definition; for more information, see activity-log. This is the default value. • **$(value)**: An inline parameter in the format **$(value)** to retrieve a value from the API context.	string

Example

```
- log:
  version: 2.0.0
  title: Gather activity log data for processing
  mode: gather-only
```

map

Use the map policy to transform your assembly flow and specify relationships between variables.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.5.0.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Map.

About

For information about the use and structure of the map policy, see The Map policy structure. For more information about API properties that affect the map policy, see API properties.

The map policy has the following format:

```
- map
   version: version
   title: title
   description: description

   inputs:
      - input_1:
          variable: context_1
          $ref: '#/definitions/definition_1'
      - input_2
          variable: context_2
          type: type_2
          content: content_type

   outputs:
      - output_3
          variable: context_3
          type: type_3

   actions:
      - set: output_3.output_property_3
        from: input_1.input_property_1

      - set: output3.output_property_3
        from:
          - input1.input_property_1
          - input2.input_property_2
        value: 'script_A'
        default: 'default_A'

      - create: output3.output_property_3
        from: input1.input_property_1
        foreach: input1.input_property_1
        actions:
          - further_actions
   options:
         .
         .
         .
    advanced_XML_and_general_configuration_options
         .
         .
         .
```

Properties

Table 2.

Property	Required	Description	Data type	Belongs to
version	Yes	The policy version number	string	N/A
title	No	A title for the policy.	String	N/A
description	No	A policy description.	String	N/A
inputs	No	An array listing the inputs of the map policy.	Object	N/A
outputs	Yes	An array listing the outputs of the map policy.	Object	N/A
variable	Yes	A reference to the context variable that is the location of the input or output variable..	String	inputs or outputs
$ref	Yes[1]	A reference to the definition of the type of the variable.	String	inputs or outputs
type	Yes[1]	The type of the variable.	String	inputs or outputs
content	No	The content type of the variable: `application/xml` or `application/json`. If None is selected, or the field is not included, then the type is treated as JSON.	String	inputs or outputs
actions	Yes	Lists actions to be performed by the map policy.	Object	N/A
set	Yes[2]	Specifies by name the output variable that is to be set to a value by the action.	String	actions
create	Yes[2]	Specifies by name the output variable that is to have a value appended to it by the action.	String	actions
from	No	Specifies by name any input variables used by the action.	String	actions
value	No	Contains a script that maps and transforms input variables into output variables.	String	actions
default	No	Contains a static value, or an inline variable reference, to be applied to the output when no input value is provided. For information on inline variable references, see Inline references.	String	actions
foreach	No	Specifies by name an array for which further actions should be performed for each element.	String	actions
includeEmptyXMLElements	No	If set to **true**, empty XML elements are included in the output of the map policy. Set to **false** if you do not want empty XML elements to be included in the output of the map policy. The default value is **true**.	Boolean	options
namespaceInheritance	No	If set to **true**, XML namespaces are inherited from the parent element. Set to **false** if you want the map policy to use explicit namespaces. The default value is **true**.	Boolean	options
inlineNamespaces	No	If set to **true**, XML namespaces will be inserted into the document where they are first used. Set to **false** if you want namespaces to all be defined on the root element. The default value is **true**.	Boolean	options
API Gateway only mapResolveXMLInputDataType	No	If set to **true**, XML elements whose schema is configured as type boolean or numeric will be converted to that data type. Set to **false** if you want all XML element values to be returned as a string.	Boolean	options

Property	Required	Description	Data type	Belongs to
API Gateway only mapParseXMLOutput	No	If set to **true**, the map policy will write XML output to **message.body** as a parsed XML document. By default, the XML will be output as an XML string. XML output to any other variable will always be written as an XML string. The default value is **false**.	Boolean	options
API Gateway only mapXMLEmptyElement	No	This property controls how the map policy handles the output of an empty XML element. Specify one of the following values: • **string** (the default option): An empty XML element is handled as an empty string. • **null**: An empty XML element is handled as a null value. • **none**: The data is ignored. • API Gateway only **string-badgerfish**: The value for an empty XML element is considered to be an empty string. The empty string value will be placed into a JSON badgerfish value property. • API Gateway only **null-badgerfish**: The value for an empty XML element is considered to be null. The null value will be placed into a JSON badgerfish value property. A mapping of this element to a JSON output property does not occur unless the **mapNullValue** property is set to **true**.	String	options
API Gateway only mapArrayFirstElementValue	No	If set to **true**, then if an array is encountered in the traversal of the input, only the first element is used. Set to **false** if you want the map policy to use all array elements. The default value is **false**.	Boolean	options
API Gateway only mapResolveApicVariables	No	If set to **true**, API Connect variable references found in the map properties are resolved. Set to **false** if you want the map policy to ignore API Connect variable references in the map policies. The default value is **false**.	Boolean	options
API Gateway only mapNullValue	No	If set to **true**, an input property value with a null value is mapped to the output document. Set to **false** if you want the map policy to ignore null input values. The default value is **false**.	Boolean	options
API Gateway only mapOptimizeSchemaDefinition	No	If set to **true**, complex schema types evaluation handles circular type references in an optimized manner. Set to **false** to evaluate these schema types in a standard manner. The default value is **false**.	Boolean	options

Property	Required	Description	Data type	Belongs to

API Gateway only
mapEmulateV4DefaultRequiredProps
	No	If set to **true**, default values are generated in the output for required properties that are either not mapped, or for which there is no input data present, in the following specific cases:
		• An array consists of objects that contain one or more required properties.
		• An object which is optional has one or more child properties that are required.
		By default, these required properties are not present in the output. If you set this property to **true**, these required properties will be present in the output. If the output schema defines a **default** property for the output property then the specified default value is used, otherwise a default value is assigned dependent on the data type, as follows:
		• String: empty string ("")
		• Number: 0
		• Boolean: false
		• Object: empty object
		• Array: empty array
		Example 1
		The input data has the following array of objects:
		`[{"a": "value1"}, {"a": "value2", "b": "value3"}]`
		The output schema defines the output object as having two properties, **a** and **b**, of which **b** is required. The map policy defines the following mappings:
		• `input.array.a` to `output.array.a`
		• `input.array.b` to `output.array.b`
		If this property is set to **true**, and **b** is either not mapped or has no input data present, then **b** is assigned a default value of an empty string, and the output is as follows:
		`[{"a": "value1", "b": ""}, {"a": "value2", "b": "value3"}]`
		Example 2
		The output schema defines the following structure:
		`{"a" : {"b" : {"c" : "value1", "d" : "value2"} } }`
		Property **b** is optional but property **d** within **b** is required.
		The map policy defines a mapping to `output.a.b.c`.
		If this property is set to **true**, and **d** is not mapped, then **d** is assigned a default value of an empty string, and the output is as follows:
		`{"a" : {"b" : {"c" : "value1", "d" : ""} } }`
		If this property is set to **false**, these required properties are **not** created in the output with their default values.
		The default value is **false**.
			Boolean	options

API Gateway only mapEnablePostProcessingJSON	No	If set to **true**, mapped JSON output is post processed. The post processing of JSON output will use the output schema to ensure that property values are of the same data type as that defined in the schema. It will also normalize output property values that have a Badgerfish JSON syntax due to object mapping of an XML input. Set to **false** if you want no post processing of mapped JSON output. The default value is **false**.	Boolean	options
API Gateway only mapCreateEmptyArray	No	This property controls how the map policy handles the output of an empty array. Specify one of the following values: • **all**: Output all empty arrays, including empty children arrays. • **parent**: Output only the current property's empty array value. Children map actions of this property are not attempted. • **none**: Prevent any empty output array values from being produced. The default value is **all**.	String	options
API Gateway only mapReferenceLimit	No	Set the value of this property to an integer value that specifies the maximum allowed number of iterations of a circular schema definition. The default value is 1, which means that circular schema definitions are not followed. The maximum possible value is 5. If you specify a value greater than 5, a value of 5 is assumed. If you specify a non-numeric value, a value of 1 is assumed.	String	options
messagesInputData	No	This property defines the severity level for log messages that relate to input data. Specify one of the following values: • **error** • **warn** • **info**	String	options

Property	Required	Description	Data type	Belongs to

API Gateway only
mapEmulateV4EmptyJSONObject
	No	If a mapping fails because its input is not present and there is no default mapping configured, the default behavior is to not to make any change to the output mapping. Set this property to **true** to create an empty object for the parent of the target mapping, emulating the behavior of IBM® API Management Version 4.0.
		
		Example
		The map policy defines a mapping to **output.a.b.c**.
		If input data is present, the output is as follows:
		`{ "a": { "b": { "c": "inputvalue" } } }`
		If there is no input data, and the mapEmulateV4EmptyJSONObject option is set to **true**, the output is as follows:
		`{ "a": { "b": { } } }`
		Properties **a** and **b** are created but the value of **b** is an empty object.
		The default value is **false**.
			Boolean	options

[1] There must be one of $ref or type in the description of a variable.

[2] There must be one of set or create in an actions field.

Example

```
    - map:
        version: 1.0.0
        title: Output mapping
        inputs:
          Monthly_cost:
            schema:
              type: double
            variable: loan_invoke.body.monthly_payment
            content: application/json
          Duration:
            schema:
              type: integer
            variable: request.parameters.duration
        outputs:
          Quote_Output:
            schema:
              $ref: '#/definitions/Quote_Output'
            variable: message.body
            content: application/json
        actions:
          - set: Quote_Output.monthly_repayment
            from: Monthly_cost
            value: ''
          - set: Quote_Output.total_cost
            from:
              - Duration
              - Monthly_cost
            value: '$(Duration)*$(Monthly_cost)'
        description: Maps and transforms contexts to the operation output.
        options:
          includeEmptyXMLElements: false
          namespaceInheritance: false
          inlineNamespaces: false
          API Gateway only mapResolveXMLInputDataType: true
          API Gateway only mapXMLEmptyElement: null
          API Gateway only mapArrayFirstElementValue: false
          API Gateway only mapResolveApicVariables: false
          API Gateway only mapNullValue: true
          API Gateway only mapOptimizeSchemaDefinition: true
          API Gateway only mapCreateEmptyArray: parent
          API Gateway only mapReferenceLimit: 5
          messagesInputData: warn
          mapEmulateV4EmptyJSONObject: true
```

operation-switch

Use the operation-switch construct when you want to execute alternative policy assemblies, conditional on the operation that is being called.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see operation-switch.

About

An operation can be described with a **verb**/**path** pair, or with an **operationId**. The **operationIds** are strings, or names, that are defined in the OpenAPI document.

The operation-switch policy has the following format:

```
- operation-switch:
  version: version
  title: title
  description: description
  case:
    - operations:
      - verb: operation_verb_1_1
        path: operation_path_1_1
      - verb: operation_verb_1_2
        path: operation_path_1_2
                      .
                      .
                      .
          further verb/path combinations
                      .
                      .
                      .
      execute:
        policy_assembly_1 ...
    - operations:
      - verb: operation_verb_2_1
        path: operation_path_2_1
      - verb: operation_verb_2_2
        path: operation_path_2_2
                      .
                      .
                      .
          further verb/path combinations
                      .
                      .
                      .
      execute:
        policy_assembly_2 ...
                    .
                    .
                    .
    - operations:
      - operationID_3_1
      - operationID_3_2
      - operationID_3_3

                      .
                      .
                      .
          further operationIDs
                      .
                      .
                      .
      execute:
        policy_assembly_3 ...
                    .
                    .
                    .
        further operations sections
                    .
                    .
                    .
```

For each **operations:** section, if the operation that is being called matches any of the **verb**/**path** combinations or **operationId** strings listed in that **operations:** section, then the policy assembly that is defined in the **execute:** section is executed. Therefore, each **operations:** section defines execution of a policy assembly conditional on the operation that is being called.

You can have as many **operations:** sections as you want, and each **operations:** section can have one or more **verb**/**path** combinations or **operationId** strings.

The **execute:** section can define any policy assembly, including further operation-switch policies.

Properties

Table 2. operation-switch policy properties

Property	Required	Description	Data type	Contained in
version	Yes	The policy version number	string	N/A
title	No	A title for the policy.	string	N/A
description	No	A policy description.	string	N/A
case	Yes	An array containing different cases, each entry contains an operations and execute field.	array (object)	N/A
operations	Yes	The operations to which a case applies.	object	case
verb	No	An operation verb. Valid values: • `GET` • `POST` • `PUT` • `DELETE` • `HEAD` • `PATCH` • `OPTIONS`	string	operations
path	No	A relative path to an individual endpoint. For example, `/account_status`.	string	operations
operationID	No	The operation is defined in OpenAPI. The policy operation refers to the OpenAPI operation.	string	operations
execute	Yes	The policy assembly that you want to execute if the operation that is being called matches any one of the verb/path combinations. For more information, see execute.	string	case

Example

Example that defines match operations with `verb/path` combinations:

```
# carry out different redaction actions depending on the operation

- operation-switch:
  version: 1.0.0
  title: clear_region_and_set_body
  case:
    - operations:
      - verb: GET
        path: /account_details
      execute:
        - redact:
            title: remove secret field
            actions:
              - action: remove
                from: all
                path: /document/user/secret
    - operations:
      - verb: GET
        path: /account_status
      execute:
        - redact:
            title: redact address
            actions:
              - action: redact
                from: response
                path: //*[@name='secondaryAddress']/*[@name='streetAddress']
```

Example that defines match operation with `operationIDs`:

```
# match on operationIDs

- operation-switch:
  title: customer_actions
  case:
    - operations:
      - getCustomerByName
      - deleteCustomer
      - addACustomer
      execute:
           .
           .
           .
```

oauth

Use the oauth policy to policy to perform OAuth processing based on defined OAuth provider settings.

Gateway support

Table 1. Table showing which gateways support this policy, and the

corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see OAuth.

About

The oauth policy has the following format:

```
- oauth:
  version: version
  title: title
  description: description
  oauth-provider-settings-ref:
            .
            .
            .
    references_to_oauth_settings
            .
            .
            .
  supported-oauth-components:
    - oauth_component_1
    - oauth_component_2
            .
            .
            .
```

Note: You add an oauth policy to the OpenAPI source in a native OAuth provider. For more information, see the following topics:

- Editing the native OAuth provider configuration using the API Editor (Cloud Manager UI)
- Editing the native OAuth provider configuration using the API Editor (API Manager UI)

Properties

Table 2. oauth policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A description of the policy.	string
`oauth-provider-settings-ref: default`	Yes	The name of an existing OAuth provider that defines the required settings.	string
`oauth-provider-settings-ref: url`	No	A URL to a document that contains serialized XML or JSON properties that defines OAuth token generate settings. URL reference takes precedence over any existing literal configuration or object reference.	string
`oauth-provider-settings-ref: literal`	No	A literal string that contains serialized XML or JSON properties that defines OAuth token generate settings. Literal configuration takes precedence over any existing object reference.	string
`supported-oauth-components: - oauth_component_1 - oauth_component_2 . . .`	Yes	Specify the OAuth components that are supported by this policy, as follows: - OAuthValidateRequest Validates the authorization request from the client. - OAuthGenerateAZCode Generates the authorization code for the client, which represents the resource owner's authorization that grants access to the requested resource. - OAuthVerifyAZCode Verifies the authorization code from the client. - OAuthVerifyRefreshToken Verifies the refresh token that is presented by the client. - OAuthCollectMetadata Collects metadata about a client for later user interaction - OAuthGenerateAccessToken Generates the access token to the client when the authorization code or refresh token is verified. - OAuthIntrospectToken Introspects the token to determine its state and, when active, its metadata.	string

Overriding default OAuth provider settings

You can use either the **`literal`** property or the **`url`** property to dynamically override any OAuth provider configuration settings to dynamically override any OAuth provider configuration settings defined by the **`default`** property.

For example, to override the access token expiration time with a value of 200 seconds, include the following configuration in either the literal string or the document at the specified URL:

```
<OAuthProviderSettings><APICAccessTokenTTL>200</APICAccessTokenTTL></OAuthProviderSettings>
```

For a list of all OAuth provider settings, refer to the **`OAuthProviderSettings`** management schema, defined in the xml-mgmt.xsd file located in the store: directory on the DataPower API Gateway.

If you are using the API Manager user interface, the connection details are determined by the API Manager URL that you open, and the user ID with which you log in. If you are using the API Designer user interface, you provide the management server details and user ID in the login window that opens when you first launch API Designer; see Logging into API Connect Designer.

oauth policy example

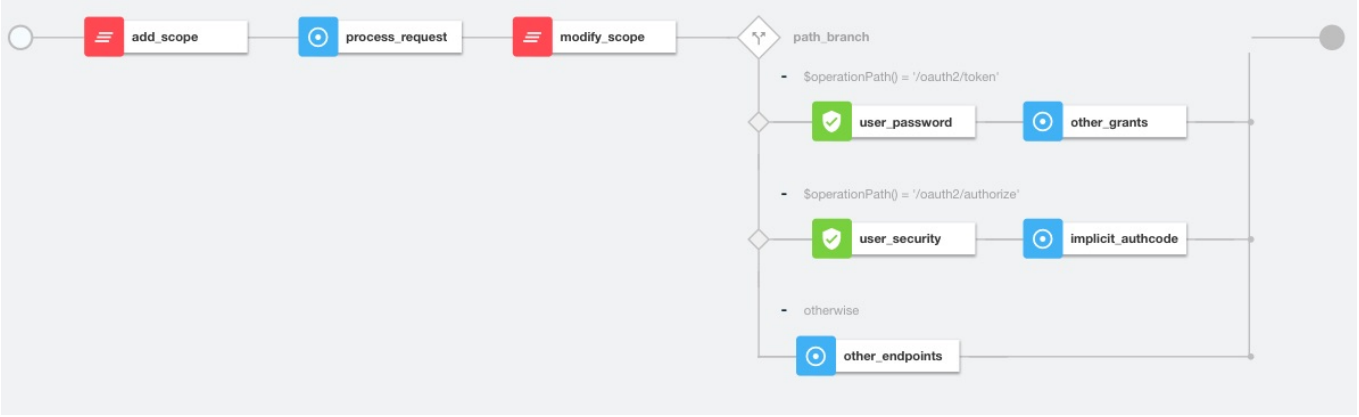
```
- oauth:
    version: 2.0.0
    title: my-oauth-policy
    oauth-provider-settings-ref:
      default: my-oauth
    supported-oauth-components:
      - OAuthGenerateAZCode
      - OAuthGenerateAccessToken
      - OAuthIntrospectToken
      - OAuthVerifyAZCode
      - OAuthVerifyRefreshToken
```

Example - using multiple OAuth policies in an OAuth provider assembly

This example demonstrates the use of multiple OAuth policies in the assembly flow for a native OAuth provider.

The example is based on the default assembly that is generated when you create a native OAuth provider, and is customized with the addition of **gatewayscript** policies that use OAuth context variables to manipulate the OAuth flow. For details on creating a native OAuth provider, see Configuring a native OAuth provider.

It has the following assembly flow:



The following sections describe the OpenAPI source code that underlies each of the policies in the assembly; for the complete assembly code, download multiple_oauth_policies.txt.

Sample policy to add a custom scope

The **add_scope** policy is a **gatewayscript** policy that adds a custom scope to the request.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
    version: 2.0.0
    title: add_scope
    source: |-
      // Add another custom scope to the request
      let scope = context.get("request.parameters.scope.values[0]");
      if (scope)
        context.set("oauth.processing.scope", scope + " custom");
```

Validate the initial OAuth request

The first **process_request** policy is an **oauth** policy that processes the initial request and verifies that the request is valid. The result of the processing is stored automatically in the oauth.processing context variables for use, as required, by the next OAuth policy in the assembly flow.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
    title: process_request
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect protocol steps
      that are needed for OAuth Validation by default. The inputs and
      outputs of each of the steps are driven by documented context
      variables. Add or remove the Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthValidateRequest
```

Sample policy to modify the scope

The **modify_scope** policy is a **gatewayscript** policy that modifies the scope depending on the calling application.

The underlying OpenAPI source YAML is as follows:

```
- gatewayscript:
    version: 2.0.0
    title: modify_scope
    source: |-
      let admin_id = '1f1a2aa4-db9f-4423-b2f1-e2572b12123a';

      // Check application and modify the scope
      let app = context.get("oauth.processing.client_id");
      let scope = context.get("oauth.processing.scope");
      if (app === admin_id) {
        context.set("oauth.processing.scope", scope + " admin");
      } else {
        context.set("oauth.processing.scope", scope + " customer");
      }
```

Branch conditionally according to the OAuth path

The **path_branch** policy is a **switch** policy that branches according to the different OAuth paths to process the resource owner.

The underlying OpenAPI source YAML is as follows:

```
- switch:
    version: 2.0.0
    title: path_branch
    case:
      - condition: ($operationPath() = '/oauth2/token')
        execute:
            .
            .
            .
          definition of the user_security and other_grants policies
            .
            .
            .
      - condition: ($operationPath() = '/oauth2/authorize')
        execute:
            .
            .
            .
          definition of the user_password and implicit_authcode policies
            .
            .
            .
      - otherwise:
            .
            .
            .
          definition of the other_endpoints policy
            .
            .
            .
```

Process the user name and password, and enable grant type component

The following two policies operate on the token endpoint.

- The **user_password** policy for password grant type processes the user name and password from the **x-www-form-urlencoded** body. The authentication method is derived from the User Security settings in the OAuth provider; see Configuring user security for a native OAuth provider.
- The **other_grants** policy is an **oauth** policy that enables the **OAuthGenerateAccessToken**, **OAuthVerifyAZCode**, **OAuthVerifyRefreshToken**, and **OAuthCollectMetadata** components to perform the operations for client credentials, authorization code, refresh token, and password grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
    title: user_password
    version: 2.0.0
    description: ''
    factor-id: default
    extract-identity-method: context-var
    user-context-var: request.parameters.username.values
    pass-context-var: request.parameters.password.values
    ei-stop-on-error: false
    user-auth-method: auth-url
    au-stop-on-error: false
    auth-url: 'http://httpbin.org/basic-auth/user/pass'
    user-az-method: authenticated
    az-stop-on-error: true
    auth-response-headers-pattern: (?)x-api*
    auth-response-header-credential: X-API-Authenticated-Credential
- oauth:
    title: other_grants
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for token path by default.
```

```
      The inputs and outputs of each of the steps are driven by
      documented context variables. Add or remove the Supported
      OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthGenerateAccessToken
      - OAuthVerifyAZCode
      - OAuthVerifyRefreshToken
      - OAuthCollectMetadata
```

Perform authorization checks, and enable grant type components

These following two policies operate on the authorize endpoint.

- The **user_security** policy configuration is derived from the User Security settings in the OAuth provider; see Configuring user security for a native OAuth provider.
- The **implicit_authcode** policy is an **oauth** policy that enables the **OAuthGenerateAZCode**, **OAuthGenerateAccessToken**, and **OAuthCollectMetadata** components to perform the operations for the implicit and authorization code grant types.

The underlying OpenAPI source YAML is as follows:

```
- user-security:
    title: user_security
    version: 2.0.0
    description: >-
      This user security policy performs EI(basic) and AU(auth
      url) check for oauth assembly. Change the security check
      method as required
    factor-id: default
    extract-identity-method: basic
    ei-stop-on-error: true
    user-auth-method: auth-url
    au-stop-on-error: true
    user-az-method: authenticated
    az-stop-on-error: true
    auth-response-headers-pattern: (?)x-api*
    auth-response-header-credential: X-API-Authenticated-Credential
    auth-url: 'http://httpbin.org/basic-auth/user/pass'
- oauth:
    title: implicit_authcode
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for az code path by
      default. The inputs and outputs of each of the steps are
      driven by documented context variables. Add or remove the
      Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthGenerateAZCode
      - OAuthGenerateAccessToken
      - OAuthCollectMetadata
```

Process all other endpoints

The **otherwise** condition catches all other endpoints such as the introspect and revoke endpoints. The **other_endpoints** policy in the **otherwise** condition is an **oauth** policy that enables the **OAuthIntrospectToken**, and **OAuthRevokeTokencomponents** components to perform the operations for introspect and revoke.

The underlying OpenAPI source YAML is as follows:

```
- oauth:
    title: other_endpoints
    version: 2.0.0
    description: >-
      This oauth policy performs all OAuth/OpenID Connect
      protocol steps that are needed for all other paths by
      default. The inputs and outputs of each of the steps are
      driven by documented context variables. Add or remove the
      Supported OAuth Components as required.
    oauth-provider-settings-ref:
      default: custom-form
    supported-oauth-components:
      - OAuthIntrospectToken
      - OAuthRevokeToken
```

Related concepts

- API policies and logic constructs

Related reference

- OAuth context variables

Related information

- Configuring a native OAuth provider

parse

Use the parse policy to control the parsing of an input document. When the input document is a JSON string, the string is parsed instead of copied over.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later) 2.2.0 (DataPower API Gateway Version 10.5.0.5 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Parse.

About parse

The parse policy has the following format:

```
- parse:
  version: version
  title: title
  description: description
  use-content-type: request_header_usage_setting
  parse-settings-reference:
            .
            .
            .
    references_to_parse_settings
            .
            .
            .
  input: input_message
  output: output_message
```

Properties

Table 2. parse policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
use-content-type	No	If this setting is enabled and the parse setting is configured to detect the document type, the parse operation uses the **Content-Type** specified in the request headers. If this setting is enabled and the document type in the parse setting is configured for either JSON or XML, the parse operation uses the **Content-Type** specified in the request headers and fails if the **Content-Type** in the request headers does not match the parse setting. Enabling this setting is applicable only when the expected **Content-Type** is either JSON or XML. If this setting is not enabled, the parse operation uses either the document type specified in the parse setting, or the detected document type if the parse setting is configured to detect the document type. The default value is **false**.	boolean
parse-settings-reference: default	No	An existing valid object from which to retrieve default property values for the dynamic object.	string

Property	Required	Description	Data type

document_type	No	The type of document to parse. Specify one of the following values:
		
		• **detect**: the document type is detected and the payload is parsed accordingly. This type is the default option.
		• **JSON**: the payload is parsed as JSON.
		• **XML**: the payload is parsed as XML.
		• **graphql**: parsing is performed in one of the following ways:
		  ◦ As GraphQL in the body of a POST request. The GraphQL is in the following format:
		
		`{ me { name } }`
		
		  ◦ As a JSON string in the body of a POST request that contains a GraphQL query. The JSON object is in the following format:
		
		`{ "query": "query MyData ($idVar: Int!) { user (id: $idVar) { name }}", "operationName": "MyData", "variables": {"idVar": 123} }`
		
		  ◦ Through the URL query of a GET request. The URL is in the following format:
		
		`https://myapi/graphql?query={me{name}}`
		
		The following example includes the **variables** and **operationName** properties, with URL encoding:
		
		`https://hostname/basepath/graphql?query=query%20fetchAccounts%20($limit:%20Int)%20{accounts(limit:%20$limit)%20{name{first,last}}}&variables={"limit":100}&operationName=fetchAccounts`
		
		Query variables in a GET request can be sent as a JSON string in a query parameter called **variables**. If the query contains several named operations, an **operationName** query parameter is required to determine which operation is to be executed.
			string

max_doc_size	No	Optionally, the maximum document size in bytes that the parse operation accepts. This setting provides threat protection by enforcing the maximum document size. Enter a value in the range 0 - 5368709121. A document is rejected when its size exceeds the maximum size. A value of 0 indicates an unlimited document size. When set to 0, the operation result does not return the document size. This setting is applicable to JSON, XML, or GraphQL input documents. The default value is 4194304.	integer
max_nesting_depth	No	Optionally, the maximum nesting depth of an XML, JSON, or GraphQL message that the parse operation accepts. This setting provides threat protection by enforcing limits in the message. • When XML, it is the maximum level of element depth. • When JSON, it is the maximum level of nested label-value pairs, the maximum number of nested arrays, or the maximum number of combination of label-value pairs and arrays. • When GraphQL, it is the maximum level of nested selection sets. Enter a value in the range 0 - 4096. A document is rejected when its nesting depth exceeds the maximum depth. A value of 0 indicates unlimited nesting depth. When set to 0, the operation result does not return the nesting depth. The default value is 512.	integer
max_width	No	Optionally, the maximum width of the payload that the parse operation accepts. • When the input document is XML, this property specifies the maximum number of attributes for an element or the maximum number of child elements for an element. • When the input document is JSON, this property specifies the maximum number of properties on a JSON object or the maximum number of JSON items in a JSON array. • When the input document is GraphQL, this property specifies the maximum number of selections in a selection set. Enter a value in the range 0 - 65535. The document is rejected when its width exceeds the maximum width. A value of 0 indicates unlimited width. When set to 0, the operation result does not return the width of the document. The default value is 4096.	integer
max_name_length	No	Optionally, the maximum name length in bytes in a document that the parse operation accepts. • For XML, it is the length of the name portion of a tag. • For JSON, it is the length of the label portion of the JSON label-value pair. • For GraphQL, it is the maximum length of the identifiers, including field names and directive names. The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 8192. A document is rejected when its name length exceeds the maximum length. A value of 0 indicates unlimited name length. When set to 0, the operation result does not return the name length of a document. The default value is 256.	integer

Property	Required	Description	Data type
max_value_length	No	Optionally, the maximum value length in bytes in a document that the parse operation accepts. • For XML, it is the length of an attribute or the length of a text value. • For JSON and GraphQL, it is the length of a string value. The length includes any white space that is contained between tags in XML or quotation marks in JSON. Enter a value in the range 0 - 5368709121. A document is rejected when its value length exceeds the maximum length. A value of 0 indicates unlimited value length. When set to 0, the operation result does not return the value length of a document. The default value is 8192.	integer
max_unique_names	No	Optionally, the maximum number of unique names in a JSON or XML document that the parse operation accepts. • For XML, it is the number of unique XML local names. • For JSON, it is the number of unique JSON labels. Enter a value in the range 0 - 1048575. A document is rejected when the number of unique names in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique names. When set to 0, the operation result does not return the number of unique names in a document. The default value is 1024.	integer
max_unique_prefixes	No	Optionally, the maximum number of unique XML namespace prefixes in a document that the parse operation accepts. This limit counts multiple prefixes defined for the same namespace, but does not count multiple namespaces defined in different parts of the input document under a single prefix. Enter a value in the range 0 - 262143. A document is rejected when the number of unique prefixes in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique prefixes. When set to 0, the operation result does not return the number of unique prefixes in a document. The default value is 1024.	integer
max_unique_namespaces	No	Optionally, the maximum number of unique XML namespace URIs that the parse operation accepts. This limit counts all XML namespaces, regardless of how many prefixes are used to declare them. Enter a value in the range 0 - 65535. A document is rejected when the number of unique namespaces in the document exceeds the maximum number. A value of 0 indicates an unlimited number of unique namespaces. When set to 0, the operation result does not return the number of unique namespaces in a document. The default value is 1024.	integer
max_number_length	No	Optionally, the maximum number of bytes in the value portion of a JSON label-value pair when the value is a number. The number must be a contiguous string of bytes that contain no white space. The number can include a minus sign and a positive or negative exponent. Enter a value in the range 0 - 256. A document is rejected when the number length in the document exceeds the maximum length. A value of 0 indicates unlimited number length. When set to 0, the operation result does not return the number length in a document. The default value is 128.	integer
strict-utf8-encoding	No	For JSON documents, whether to enforce strict UTF-8 encoding throughout the entire document. • When **on**, the entire document is checked for valid UTF-8 encoding. • When **off**, only the first few bytes are checked for proper encoding. The rest of the document is assumed to be in the same encoding. The default value is **off**.	boolean
parse-settings-reference: literal	No	A literal string as serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing default properties.	string
parse-settings-reference: url	No	A URL that represents a named context from which to retrieve the serialized XML or JSON properties that are merged into the dynamic object. These properties take preference over any existing literal or default properties.	string
input	No	The name of a variable in the API context. The content of the **body** field of the variable is the input to the policy. The default variable name is **message**.	string
output	No	The name of a variable in the API context. The content of the body field of the variable is the output of the parse operation. The parse metrics of the parsed document can be stored in a different part of the message. The default variable name is the same as the input name, so by default the input message is overwritten by the output message.	string

parse policy example

```
- parse:
    version: 2.0.0
    title: my-parse-policy
    use-content-type: true
    parse-settings-reference:
      default: my-parse
    input: input-message
    output: output-message
```

parse-settings-reference: literal example

In XML:

```
- parse:
    version: 2.0.0
```

```
    title: parse
    parse-settings-reference:
    default: apic-default-parsesettings
    literal: <ParseSettings><DocumentType>xml</DocumentType></ParseSettings>
```

In JSON:

```
- parse:
    version: 2.0.0
    title: parse
    parse-settings-reference:
    default: apic-default-parsesettings
    literal: { \"ParseSettings\" : { \"propertyName\" : propertyValue } }
```

proxy

Apply the proxy policy to proxy another API within your operation, particularly if you need to call a large payload.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway, functionality provided by Invoke	

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Proxy.

About

Keep the following considerations in mind regarding the proxy policy.

- Only one proxy policy is permitted to be called per assembly.
- More than one proxy policy can be applied, if they are contained in mutually exclusive branches of the assembly.
- You can use the proxy policy to return multipart form data, that is, when the response is set to `Content-Type: multipart/related`. However, proxy must be the final policy in the assembly, otherwise the response that is received is manipulated causing the multipart form data to be lost.
- The proxy policy, if inside a conditional policy, must be the **final** policy to be executed in the API. If you need further processing afterward, use the invoke policy rather than the proxy policy.
- The Proxy policy does not currently attempt to rewrite a Location header that is returned from the back end.

The proxy policy has the following structure:

```
- proxy:
  version: version
  title: title
  description: description
  target-url: URL_of_target_API
  tls-profile: TLS_profile_to_be_used
  verb: method_type
  http-version: HTTP_version
  timeout: timeout_value_in_seconds
  compression: is_data_to_be_compressed
  username: username_if_authentication_required
  password: password_if_authentication_required
  output: location_of_the_proxy_result
  cache-key: unique_identifier_of_the_document_cache_entry
  cache-response: cache_behavior
  cache-putpost-response: response_caching_behavior
  cache-ttl: cache_time_to_live
  stop-on-error: errors_that_stop_the_flow
```

Properties

The table describes the properties of the proxy policy.

Table 2. Proxy policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
target-url	Yes	The URL of the target API.	string
tls-profile	No	The TLS profile to be used.	string

Property	Required	Description	Data type
verb	No	The operation method type. Valid values: • Keep • GET • POST • PUT • DELETE • PATCH • HEAD • OPTIONS The default value is **Keep**.	string
http-version	No	The HTTP version. The default value is **1.1**.	string
timeout	No	The timeout value in seconds. The default value is **60**.	integer
compression	No	Specifies whether data is to be compressed by using gzip before it is uploaded. The default value is **false**.	boolean
username	No	The user name, if authentication is required.	string
password	No	The password, if authentication is required.	string
output	No	Specifies the location of the proxy result. By default, the proxy result, that is the body, headers, statusCode and statusMessage, is saved in the variable **context.message**. The assembly developers can specify an additional location to store the proxy result with the output property.	string
cache-key	No	Specifies the unique identifier of the document cache entry.	string
cache-response	No	The cache response type. Valid values: • protocol: The cache behavior is defined by the Cache-Control headers on the request and response. • no-cache: Specifies that there is no caching. However, if the document is already in the cache, the document is retrieved from the cache. • time-to-live: Specifies that the response stays in the cache for the specified time. The default value is **protocol**.	string
cache-putpost-response	No	Specifies whether to cache the response from POST and PUT requests. Caching the response from POST and PUT requests can reduce server load and reduce latency in the response to the client request. The default value is **false**.	boolean
cache-ttl	No	Specifies the amount of time in seconds that the response stays in the cache. Applies only if the property cache-response is set to **time-to-live**. Enter a value in the range 5 - 31708800. The default value is **900**.	integer
stop-on-error	No	List the errors that, if thrown during the policy execution, cause the flow to stop. If there is a **catch** flow configured for the error, it is triggered to handle the error thrown. If an error is thrown and there are no errors specified for the Stop on error property, or if the error thrown is not one of the specified errors, the policy execution is allowed to complete, and the assembly flow continues.	string

API Gateway only

ratelimit

Use the ratelimit policy to apply one or more rate, burst, or count limits at any point in your API assembly flow. Rate and burst limits restrict the number of calls made to an API in a specified time period, while count limits impose a strict limit on the total number of calls.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.0 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.5.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Rate Limit.

About

The defined rate, burst, and count limits are applied to whatever follows in the assembly flow. For example, if a ratelimit policy is placed before an invoke policy, and the call made by the invoke policy exceeds the limits defined by the ratelimit policy, the API call itself fails.

The ratelimit policy has the following format:

```
- ratelimit:
    version: version
    title: title
    description: description
    source: rate_and_burst_limit_location
```

```
    rate-limit: rate_limits_to_apply
    burst-limit: burst_limits_to_apply
```

Properties

The following table describes the properties of the ratelimit policy.

Table 2. ratelimit policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
source	Yes	The location of all the rate limit, burst limit, and count limit definitions that are included in this policy. Specify one of the following values: • **`catalog-named`**: the rate or burst limits to be applied are defined in the appropriate **`api-collection`** object on the DataPower API Gateway, which is the object that represents your API Connect Catalog in the gateway configuration. For details of how to configure a rate, burst, or count limit in the **`api-collection`** object, see Configuring a rate, burst, or count limit on the DataPower API Gateway. • **`plan-named`**: the limits to be applied are assembly burst limits or assembly count limits defined on the Plan to which the calling application is subscribed. For details on how to configure assembly burst limits and assembly count limits in a Plan, see Editing a draft Product. Note: The following conditions apply to a **`plan-named`** rate limit policy: ◦ Every Plan that contains the API that is being limited must have the named rate limit applied. ◦ A **`plan-named`** rate limit policy cannot be used with automatically generated Products. • **`gateway-named`**: the limits to be applied are defined in the **`apigw`** object, named **`apiconnect`**, on the DataPower API Gateway gateway. For details of how to configure a rate, burst, or count limit in the **`apigw`** object, see Configuring a rate, burst, or count limit on the DataPower API Gateway. • **`plan-default`**: the rate and burst limits that are applied are the default ones configured in the Plan to which the calling application is subscribed. For details on how to configure default Plan rate and burst limits, see Editing a draft Product. Note: The use of this option does not disable any automatic preflow rate limit policy that might be in place, and could result in the application of the rate limiting being repeated. For more information on preflow policies, see Customizing the preflow policies.	string
rate-limit	Yes[1]	If the **`source`** property is set to **`catalog-named`**, a list of rate limit names as defined in the DataPower API Gateway configuration, [1]You must provide at least one rate, burst, or count limit.	string
burst-limit	Yes[2]	If the **`source`** property is set to **`catalog-named`**, a list of a burst limit names as defined in the DataPower API Gateway configuration. If **`source`** is set to **`plan-named`**, a list of names of assembly burst limits defined in a Plan. [2]You must provide at least one rate, burst, or count limit.	string
count-limit	Yes[3]	If **`source`** property is set to **`catalog-named`**, a list of count limit names as defined in the DataPower API Gateway configuration. If **`source`** is set to **`plan-named`**, a list of names of assembly count limits defined in a Plan. [3]You must provide at least one rate, burst, or count limit..	string

Property	Required	Description	Data type
operation	No	For a rate limit, specify one of the following values: • **consume**: the policy reduces the balance of available requests that remain in the interval defined by the rate limit specified by the **rate-limit** property. The amount by which the remaining balance is reduced is the value that results from the weight expression defined in the specified rate limit. • **replenish**: the policy increases the balance of available requests that remain in the interval defined by the rate limit specified by the **rate-limit** property. The amount by which the remaining balance is increased is the value that results from the weight expression defined in the specified rate limit. The purpose of the replenish operation is to restore an appropriate amount of remaining balance in cases where the amount calculated by an earlier consume operation turns out to be too high. For example, for an assembly rate limit with a limit of 100, a consume operation with a weight of 60 uses 60 requests, leaving 40 requests remaining for the interval. If the weight expression in the subsequent replenish operation computes to 15, meaning that the weight expression calculated that the actual request usage was only 45, the replenish operation increases the remaining requests by 15, to a remaining balance of 55. Note that the replenish operation cannot increase the remaining requests allowed beyond the original limit defined in the assembly rate limit scheme. In this example, a replenish operation cannot replenish requests beyond 100. A replenish operation is skipped if the weight expression evaluates to a value of less than 1. The default value is **consume**. For a count limit, specify one of the following values: • **inc**: the current count is incremented by the value that results from the weight expression defined in the count limit specified by the **count-limit** property. • **dec**: the current count is decremented by the value that results from the weight expression defined in the count limit specified in the **count-limit** field. By default, the count limit is automatically decremented at the end of the API assembly by the total amount incremented during the assembly, minus any decrement operations that are defined in the assembly. You do not need to explicitly define decrement operations unless you want the decrements to occur at specific points in the assembly. However, if the auto decrement feature has been explicitly disabled in the count limit configuration, you must include a corresponding decrement operation, otherwise when the count limit is reached all future calls are blocked. For more information on configuring a count limit, see Configuring a rate, burst, or count limit on the DataPower API Gateway. The default value is **inc**. For a burst limit, the only possible value is **consume**, meaning that the limit applied is that defined in the burst limit specified by the **burst-limit** field.	string

Example

```
- ratelimit:
    version: 2.0.0
    title: Apply rate and burst limits
    source: catalog-named
    rate-limit:
      - 30perMinute
      - 2000per3Hours
    burst-limit:
      - 5per10Seconds
```

API Gateway

redact - DataPower API Gateway

Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

Gateway support

Note: This page describes the redact policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see redact - DataPower Gateway (v5 compatible).
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version

Gateway	Policy version
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Redaction - DataPower API Gateway.

About

The redact policy has the following format:

```
- redact:
    version: version
    title: title
    description: description
    redactions:
      - action: remove_or_redact
        path: JSONata_expression_for_field_to_remove_or_redact
                .
                .
                .
    root: content_source
```

Note: With the DataPower API Gateway, the input to the redact policy must be parsed data. One way to produce parsed data is to use a **parse** policy before a redact policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table describes the properties of the redact policy.

Table 2. redact policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
root	No	Specifies the data source that contains the content to which the redact or remove action applies. If the root property is omitted, the action is applied to the entire API context. You can use any supported JSONata path expression. If you want to apply the action to either request or response data, specify a value of `message.body`. The actual content to which the action is applied then depends on the positioning of the redact policy in the overall assembly flow; for example: • If positioned at the beginning, the action is applied to the client request. • If positioned after an invoke policy, the action is applied to the response from the back end. • If positioned at the end, the action is applied to the response that is returned to the client. If, in your assembly flow, the redact policy is used after a log policy that specifies **`gather-only`** for the mode property, specify a root value of **`log.request_body`** for the logged request payload, or **`log.response_body`** for the logged response payload.	string
path	Yes	Specifies a JSONata path expression that identifies the fields to redact or remove from the source. For more information, see Constructing JSONata expressions to redact fields	string
action	Yes	Specifies whether you want to remove or redact the content. Supply one of the following values: • **`remove`**: Completely removes the specified fields. • **`redact`**: Redacts (obfuscates with "*"s) the fields to block out the data. The default value is **`redact`**. Note: If a numerical value is being redacted, the redacted value is depicted as **`******`** and the type is changed to **`string`**.	string

Example

```
# Specify separate remove and redact actions

- redact:
    version: 2.0.0
    title: remove price, redact author
    redactions:
      - action: remove
        path: xpath($, ''//price'')"
      - action: redact
        path: $.**.author"
    root: message.body
```

DataPower Gateway (v5 compatible)

redact - DataPower Gateway (v5 compatible)

Use the redact policy to completely remove or to redact specified fields from the request body, the response body, and the activity logs. You might find this policy useful for removing or blocking out sensitive data (for example, credit card details) for legal, security, or other reasons.

Gateway support

Note: This page describes the redact policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see redact - DataPower API Gateway.
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Redaction - DataPower Gateway (v5 compatible).

About

The redaction policy has the following format:

```
- redact:
  version: version
  title: title
  description: description
  actions:
    - action: remove_or_redact
      from:
        - where_the_redaction_is_to_be_applied
      path: XPath_expression_for_field_to_remove_or_redact
                    .
                    .
                    .
          further action/from/path combinations
                    .
                    .
                    .
```

You can specify as many **action**/**from**/**path** combinations as you want.

Properties

The following table describes the policy properties:

Table 2. redact policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
action	No	Specifies whether you want to remove or redact the fields. Valid values: • **remove**: Completely removes the specified fields. • **redact**: Redacts (obfuscates with "*"s) the fields to block out the data. The default value is **redact**. Note: If a numerical value is being redacted, the redacted value is depicted as ****** and the type is changed to **string**.	string
from	No	Determines where the redaction is to be applied. Valid values: • **all**: Apply the redaction to the request body, the response body, and the activity logs. • **request**: Apply the redaction to the request body only. • **response**: Apply the redaction to the response body only. • **logs**: Apply the redaction to the activity logs only. You can supply one or more values. The default value is **all**.	string
path	Yes	Specifies an XPath expression that defines the fields to remove or redact. You can construct an XPath expression that is based on JSON or XML depending on whether your API requests and responses use a JSON or an XML format. If the payload is JSON, use the DataPower XML representation of the JSON content (JSONx) to construct the expression. Note: Use a JSONx representation only to identify the XPath expressions for the fields to remove or redact. Do not change the format of any response bodies in API Manager. To learn more about constructing XPath expressions that are based on JSON or XML, see Constructing XPath expressions to redact fields.	string

Example

```
# Specify separate remove and redact actions

- redact:
  version: 1.0.0
  title: remove secret field, redact address
  actions:
    - action: remove
      from:
        - all
      path: /document/user/secret
    - action: redact
      from:
        - request
        - response
      path: //*[@name='secondaryAddress']/*[@name='streetAddress']
```

set-variable

Use the set-variable policy to set the value of a runtime variable, or to add or clear a runtime variable.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Set Variable.

About

The set-variable policy has the following format:

```
- set-variable:
  version: version
  title: title
  description: description
  actions:
    - action_type: variable_name
      value: value
      type: data_type
```

Properties

Table 2. set-variable policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
actions	Yes	Lists the actions to be performed by the set-variable policy.	array
set	Yes[1]	For setting a variable. specifies the name of the variable that you want to set. [1]One of the properties set, add, or clear is required.	string
add	Yes[2]	For adding a variable. specifies the name of the variable that you want to add. [2]One of the properties set, add, or clear is required.	string
clear	Yes[3]	For clearing a variable. specifies the name of the variable that you want to clear. [3]One of the properties set, add, or clear is required.	string
value	Yes[4]	Allocates this value to the specified variable. Can be a literal value, or another variable. [4]value is required only when **`set`** or **`add`** is specified as the action.	string
API Gateway only type	Yes	Specifies the data type of the variable. Valid values: • **`any`** • **`string`** • **`number`** • **`boolean`** For all values other than **`any`**, the value is validated against the specified data type. If the data type is specified as **`boolean`**, the value property must be set to **`true`** or **`false`**.	string

Example 1

```
# clear a variable

set-variable:
  version: 1.0.0
```

```
  title: clear_region
  actions:
    - clear: message.headers.region
```

Example 2

```
# set a variable to the value of an API Gateway context variable

set-variable:
  version: 2.0.0
  title: set content type
  actions:
    - set: message.headers.contenttype
      value: $(message.headers.content-type)
      type: string
```

Example 3

```
# add a variable

assembly:
    execute:
      - set-variable:
          version: 2.0.0
          title: set-variable
          actions:
            - value: testing add
              add: message.headers.jja
              type: string
```

switch

Use the switch component to execute one of a number of sections of the assembly based on which specified condition is fulfilled.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see switch.

About

The switch policy has the following format:

```
- switch:
  version: version
  title: switch
  description: 'Description'
  case:
    - condition: Script_1
      execute:
      Assembly_Section_1
    - condition: Script_2
      execute:
      Assembly_Section_2
    - otherwise:
      Assembly_Section_3
```

The **execute:** section can define any policy assembly, including further switch policies. For more information, see execute.

Properties

Table 2. switch policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
case	Yes	Contains the condition and execute pairs of the switch policy.	string

Property	Required	Description	Data type
condition	Yes (one or more)	A script that returns **`true`** or **`false`**. API Gateway Use the JSONata expression language to define your condition. See Writing switch condition scripts - DataPower API Gateway DataPower Gateway (v5 compatible) Use GatewayScript to define your condition. See Writing switch condition scripts - DataPower Gateway (v5 compatible).	string
execute	Yes (one per condition)	The policy assembly that you want to execute if the condition returns **`true`**. For more information, see execute.	string
otherwise	No	The case you want to execute if no other cases are fulfilled. It functions in the same manner as an execute property. For more information, see execute.	string

For examples, see one or other of the following topics, depending on which gateway type you are using:

- Writing switch condition scripts - DataPower API Gateway
- Writing switch condition scripts - DataPower Gateway (v5 compatible)

API Gateway only

Writing switch condition scripts - DataPower API Gateway

You write condition scripts for the switch policy in the DataPower® API Gateway by using the JSONata expression language.

The payload of the message must be parsed before you can use JSONata in a switch policy condition. Therefore, a parse policy must precede the switch policy. You can then use a JSONata condition on the parsed content that is stored in the **`body`** field of the output variable. The default variable is **`message`**. For example, if the output variable is **`message`**, you can use the JSONata condition **`$exists(message.body.username)`** on the parsed content.

The following JSONata functions are supported:

- Aggregation functions:
  - **`$average(array)`**
  - **`$max(array)`**
  - **`$min(array)`**
  - **`$sum(array)`**
- Array functions:
  - **`$append(array1, array2)`**
  - **`$count(array)`**
  - **`$reverse(array)`**
  - **`$sort(array [, function])`**
  - **`$zip(array1, ...)`**
- Boolean functions:
  - **`$boolean(arg)`**
  - **`$exists(arg)`**
  - **`$not(arg)`**
- Numeric functions:
  - **`$abs(number)`**
  - **`$ceil(number)`**
  - **`$floor(number)`**
  - **`$formatBase(number [, radix])`**
  - **`$number(arg)`**
  - **`$power(base, exponent)`**
  - **`$round(number [, precision])`**
  - **`$sqrt(number)`**
- Object functions:
  - **`$keys(object)`**
  - **`$lookup(object, key)`**
  - **`$merge(array<object>)`**
  - **`$spread(object)`**
  - **`$type(value)`**
    In addition to the standard values returned by **`$type()`**, the API gateway implementation of **`$type()`** can return the following values: **`binary`**, **`empty`**, **`graphql`**, **`stream`**, or **`xml`**.
- String functions:
  - **`$contains(str, pattern)`**
  - **`$join(array[, separator])`**
  - **`$length(str)`**
  - **`$lowercase(str)`**
  - **`$match(str, pattern [, limit])`**
  - **`$pad(str, width [, char])`**
  - **`$replace(str, pattern, replacement [, limit])`**
  - **`$split(str, separator [, limit])`**
  - **`$string(arg)`**
  - **`$substring(str, start[, length])`**
  - **`$substringAfter(str, chars)`**
  - **`$substringBefore(str, chars)`**
  - **`$trim(str)`**
  - **`$uppercase(str)`**

- You can use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Table 1. Functional extensions to JSONata

Extension	Variable	Description

`$apiCtx()`	Generic access to an API context	The `$apiCtx()` extension gives generic access to an API context.
		○ Sample transform field in an extract action:
		`"$apiCtx().request.uri"`
		○ Sample condition of a case in a switch action:
		`"$apiCtx().request.path = '/simple/apictx-function'"`

`$header(name)`	`message.headers.name`	Message header
`$httpVerb()`	`request.verb`	HTTP method of the request
`$operationID()`	`api.operation.id`	ID of the operation
`$operationPath()`	`api.operation.path`	Path of the operation
`$queryParameter('name')`	○ `request.parameters.`*`name`*`.locations` The supported keyword is `query`. ○ `request.parameters.`*`name`*`.values`	Searches for the index of `query` in `request.parameters.`*`name`*`.locations` and returns `request.parameters.`*`name`*`.values[index]`, where `[index]` is the value for `query` in locations. Parameter values are not URL decoded.
`$statusCode()`	`message.status.code`	Status code
`$storageType([arg])`	*`variable`*`.body` You can specify any variable in the API context. When no variable is specified, the default variable `message.body` is used.	Storage type of the message. The supported values are `binary`, `empty`, `graphql`, `json`, `stream`, or `xml`.

`$urlParameter('name')`	○ `request.parameters.`*`name`*`.locations` The supported keywords are `path` and `query`
	○ `request.parameters.`*`name`*`.values`
		Searches for the index of `path` and `query` in `request.parameters.`*`name`*`.locations` and returns a single array that contains both `path` and `query` values from `request.parameters.`*`name`*`.values`. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded.
		For example, the following URL contains both path and query parameter values.
		`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`
		The `$urlParameter('breed')` URL returns the following array of values.
		`[cats, adopt, Sphynx, Siamese]`
		In this example, the URL includes an API path that is configured as `/petstore/{breed}/{breed}`, where `breed` is configured to be a path parameter of the API path. As a result, `cats` and `adopt` are included in the output.

`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The `xpathExpression` must be a literal string.	Allows use of XPath expressions. The following example specifies all price elements in the source. `$xpath($, '//price')`

Table 2. Functional extensions to JSONata for GraphQL

Extension	Variable	Description
`$gqlActiveOperation([graphql_message])`	`message.body`	Gets the active operation found in the specified GraphQL message. The `operationName` must be the same as the name of the active operation.
`$gqlAlias(graphql_field_node)`	`message.body`	Gets the alias of a GraphQL field node.
`$gqlFragments([graphql_message])`	`message.body`	Gets the fragments found in the specified GraphQL message.
`$gqlName([graphql_node])`	`message.body`	Gets the node name. For operations, the node name is the `operationName`. For fields, fragment definitions, arguments, and other elements, the node name is the name of the element. By default, the `operationName` of `message.body` is retrieved.
`$gqlOperations([graphql_message])`	`message.body`	Gets the operations found in the specified GraphQL message.
`$gqlType([graphql_node])`	`message.body`	The operation type of the active operation is retrieved for Query, Mutation, and Subscription query types.

- You can use the `&` (concatenation) navigation operator.

You can use the following JSONata numeric operators:

- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- / (division)
- % (modulo)

You can use the following JSONata comparison operators for number values or strings:

- `=`

- `!=`
- `<`
- `>`
- `<=`
- `>=`

You can also use the following operators and expressions.

- Parentheses to convert a sequence into an array, specify operator precedence, or compute complex expressions on a context value.
- Array ranges and predicate expressions.
- Single asterisk (`*`) and double asterisk (`**`) wildcard characters.

The following elements of a GraphQL query can be exposed in JSONata notation using the syntax shown.

- **`query`**
  The entire GraphQL query including operations and fragments.

- ***`operationName`***
  For an anonymous operation, ***`operationName`*** can be empty.

- ***`~fragmentSpreadName`***
- **`on~`*`typeCondition`***
- ***`~~fragmentDefinitionName`***

You can also use the following functional extensions to standard JSONata notation. Each extension corresponds to a part of the API context.

Extension	Variable	Description
`$header(name)`	**`message.headers.name`**	Message header
`$httpVerb()`	**`request.verb`**	HTTP method of the request
`$operationID()`	**`api.operation.id`**	ID of the operation
`$operationPath()`	**`api.operation.path`**	Path of the operation
`$queryParameter('`*`name`*`')`	• **`request.parameters.`*`name`*`.locations`** • **`request.parameters.`*`name`*`.values`**	Returns the value of a request query parameter given the parameter name. The function searches the **`request.parameters.`*`name`*`.locations`** array for the index position of the value **`query`**, and returns **`request.parameters.`*`name`*`.values[index]`**, where **`index`** is the index position. Parameter values are not URL decoded.
`$statusCode()`	**`message.status.code`**	Status code
`$storageType([arg])`	***`variable`*`.body`** You can specify any variable in the API context. When no variable is specified, the default variable **`message.body`** is used.	Storage type of the message. The supported values are: • **`binary`** • **`json`** • **`stream`** • **`xml`**
`$urlParameter('`*`name`*`')`	• **`request.parameters.`*`name`*`.locations`** • **`request.parameters.`*`name`*`.values`**	Given a request parameter name, returns the parameter values for all occurrences of that parameter as a path parameter or query parameter. The function searches the **`request.parameters.`*`name`*`.locations`** array for the index positions of the values **`path`** and **`query`**, and returns, in a single array, the **`request.parameters.`*`name`*`.values[index]`** values for all identified index positions. When the URL contains both path and query parameter values, the array includes the path values first followed by the query values. The values of each parameter type are added in the order that they are received. Parameter values are URL decoded. For example, the following URL contains both path and query parameter values: **`http://example.com/petstore/cats/adopt?breed=Sphynx&breed=Siamese`** The function call **`$urlParameter('breed')`** returns the following array of values: **`[cats, adopt, Sphynx, Siamese]`** In this example, the URL includes an API path that is configured as **`/petstore/{breed}/{breed}`**, where **`breed`** is configured to be a path parameter of the API path. As a result, **`cats`** and **`adopt`** are included in the output.
`$xpath(path, xpathExpression)`	You can specify any writable variable in the API context. The **`xpathExpression`** must be a literal string.	Allows use of XPath expressions

Simple condition statements

The following examples show conditions that use a single function.

This example uses the **`$httpVerb()`** extension to specify the HTTP method of the request.

```
$httpVerb()="GET"
```

This example uses the **`$operationPath()`** extension to specify the path of the operation.

```
$operationPath()="/base/path-2"
```

This example uses the **$operationID()** extension to specify the operation ID.

```
$operationID()="test-gatewayscript-GET"
```

This example uses the **$statusCode()** extension to specify the message status code.

```
$statusCode()=200
```

This example uses the **$header(name)** extension to specify the content type of the message header.

```
$header("Content-Type")="application/json"
```

Combining conditions with logical operators

You can use **and** and **or** operators to combine multiple functions in a single condition.

This example specifies an HTTP GET request and an API operation path equal to **test-gatewayscript-GET**.

```
$httpVerb()="GET" and $operationPath()="test-gatewayscript-GET"
```

This example specifies either a **POST** or **PUT** request.

```
$httpVerb()="POST" or $httpVerb()="PUT"
```

This example specifies an API operation ID equal to **test-gatewayscript_POST** and a message status code equal to **200**, or an API operation ID equal to **test-gatewayscript-GET** and a message status code equal to **500**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) or ($operationID()="test-gatewayscript-GET" and
$statusCode()=500)
```

This example specifies an API operation ID equal to **test-gatewayscript-POST** and a message status code equal to **200** with an API operation path equal to **/base/path-2**.

```
($operationID()="test-gatewayscript-POST" and $statusCode()=200) and $operationPath()="/base/path-2"
```

This example specifies a message header content type equal to **text/plain** and a message header length equal to **300**, or a message status code equal to **200**.

```
($header("Content-Type")="text/plain" and $header("Content-Length")=300) or $statusCode()= 200
```

Example switch policy

```
- switch:
  version: 2.0.0
  title: switch
  case:
    - condition: ($statusCode() = 200)
      execute:
        - invoke:
          title: invoke
          timeout: 60
          verb: GET
          cache-response: protocol
          cache-ttl: 900
          target-url: 'https://example.com/1'
    - condition: ($statusCode() != 200 and ($httpVerb() = 'GET' or $httpVerb() = 'PUT'))
      execute:
        - invoke:
          title: invoke
          timeout: 60
          verb: GET
          cache-response: protocol
          cache-ttl: 900
          target-url: 'https://example.com/2'
    - otherwise:
      - set-variable:
        title: set-variable
        actions:
          - set: message.body
            value: Default result
            description: 'Set the default result for the otherwise case'
```

DataPower Gateway (v5 compatible) only

Writing switch condition scripts - DataPower Gateway (v5 compatible)

You write condition scripts for the **switch** policy in the DataPower® Gateway (v5 compatible) by using GatewayScript.

To reference variables in your **switch** conditions, use the form **apim.getvariable('*context.location.variable*')**, where *context* is the context that you want to reference, *location* is the location of the variable within that context, and *variable* is the name of the variable.

to define a **switch** condition based on the operation called, use one of the following forms:

```
- condition: "((request.verb==='GET')&&(api.operation.path==='/path-1'))"
```

```
- condition: "((api.operation.id=='Operation_ID'))"
```

where the context variables `request.verb` and `api.operation.path` retrieve the HTTP verb and the path segment of the API and operation that you want your case to apply to, while `api.operation.id` retrieves the operation ID of your operation, if one has been specified.
For information about the context variables you can use and how to reference them in your script, see API Connect context variables.

Example switch policy

```
- switch:
  version: 1.0.0
  title: switch
  case:
    - condition: message.status.code===200
      execute:
        - invoke:
          title: invoke
          timeout: 60
          verb: GET
          cache-response: protocol
          cache-ttl: 900
          target-url: 'https://example.com/1'
    - condition: ((request.verb==='GET')&&(api.operation.path==='/path-2'))||((request.verb==='GET')&&
(api.operation.path==='/path-1'))
      execute:
        - invoke:
          title: invoke
          timeout: 60
          verb: GET
          cache-response: protocol
          cache-ttl: 900
          target-url: 'https://example.com/2'
    - otherwise:
      - set-variable:
        title: set-variable
        actions:
          - set: message.body
            value: Default result
            description: Set the default result for the otherwise case
```

throw

Use the throw construct to throw an error when it is reached during an assembly flow, usually as a result of a condition being reached.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.3.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see throw.

About

The throw policy has the following format:

```
- throw:
   title: title
   name: 'error_name'
   error-status-code: http_status_code_for_error
   error-status-reason: http_reason_phrase_for_error
   message: error_message
```

When the throw policy is encountered, the specified error and error message is returned. If a catch has been configured that the error produced by the throw policy fulfills, the catch will be triggered.

Properties

Table 2. throw policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
name	Yes	The name of the error to be thrown.	string
API Gateway only error-status-code (policy version 2.1.0 and later)	No	Specify the HTTP status code for the error. You can use the `$(variable)` format to reference the `message.status.code` API context variable.	integer
API Gateway only error-status-reason (policy version 2.1.0 and later)	No	Specify the HTTP reason phrase for the error. You can use the `$(variable)` format to reference the `message.status.reason` API context variable.	string
message	No	The message to accompany the error.	string

Example

```
      - throw:
          version: 1.0.0
          title: throw
          name: '404'
          message: Not found
```

User-defined policies

You can apply your own user-defined policies to your APIs.

The policy configuration that you add to the OpenAPI definition file must conform to the schema that you defined in the YAML file that describes the policy, and will have the following general format:

```
policy_name:
  property1: value1
  property2: value2
       .
       .
       .
```

For details on user-defined policies, and how to describe and implement them, see Authoring policies.

user-security

Use the user-security policy to extract a user's credentials, authenticate those credentials, and obtain authorization from the user.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see User Security.

About

The user-security policy has the following format:

```
- user-security:
  version: version
  title: title
  description: description
  factor-id: factor ID
  extract-identity-method: method_used_to_extract_credentials
        .
        .
        .
     properties_specific_to_the specified_identity_extraction_method
        .
        .
        .
  user-auth-method: authentication_method
        .
        .
        .
     properties_specific_to_the specified_authentication_method
        .
        .
        .
  user-az-method: authorization_method
        .
        .
        .
     properties_specific_to_the specified_authorization_method
        .
        .
        .
```

Properties

Table 2. user-security policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	The title of the policy.	string

Property	Required	Description	Data type
description	No	A description of the policy.	string
factor-id	No	The identity that identifies the results of factor-authentication in the API context.	string
extract-identity-method	Yes	Select the method that is used to extract the user credentials. The following options are available: basic Use basic authentication; no additional configuration is required. context-var The credentials are provided by API Connect context variables; specify the following properties: • user-context-var: the context variable that is used to obtain the user name. • pass-context-var: the context variable that is used to obtain the password. html-form Use forms based identity-extraction. You can use the default form or a custom form. To use the default form, specify **`ei-default-form: true`**. To use a custom form, specify **`ei-default-form: false`** and supply the following properties: • ei-custom-form: the location of the form. • ei-custom-form-tls-client-profile: the TLS client profile that is used to secure the connection to the remote server. Use the ei-form-time-limit property to specify the time allowed to submit the form. redirect Use a redirect for identity-extraction; specify the following properties: • redirect-url: the URL fragment to which to redirect the request to obtain user credentials. • redirect-time-limit: the time allowed for the transaction to complete. disabled Identity-extraction is disabled; this aspect of processing is skipped. Specify **`ei-stop-on-error: true`** to halt assembly processing in the event of identity-extraction failure. This field is required if identity extraction is not disabled.	string
user-auth-method	Yes	Select the authentication method. The following options are available: auth-url The credentials are authenticated by an external endpoint; specify the following properties: • auth-url: the URL of the authentication endpoint. • auth-tls-client-profile: the TLS client profile that is used to secure the connection to the authentication endpoint. • auth-response-headers-pattern: the pattern that is used to select which response headers to add to the API context. • auth-response-header-credential: the response header that contains the authenticated user credentials. ldap The credentials are authenticated by an LDAP user registry; use the ldap-registry property to specify the required registry. disabled Authentication is disabled; this aspect of processing is skipped. Specify **`au-stop-on-error: true`** to halt assembly processing in the event of user authentication failure. This field is required if authentication is not disabled.	string

Property	Required	Description	Data type
user-az-method	Yes	Select the authorization method. The following options are available: authenticated Implicitly accept any previously authenticated users; no additional configuration is required. html-form The user provides authorization through an HTML form. You can use the default form or a custom form. To use the default form, specify **`az-default-form: true`**. To use a custom form, specify **`az-default-form: false`**. Supply the following properties: • az-table-dynamic-entries: the name of a context variable that specifies the scopes that are to be added automatically to the authorization consent form. • az-form-time-limit: the time allowed to submit the form. For a custom form, supply the following properties: • az-custom-form: the location of the form. • az-custom-form-tls-client-profile: the TLS client profile that is used to secure the connection to the remote server. disabled Authorization is disabled; this aspect of processing is skipped. Specify **`az-stop-on-error: true`** to halt assembly processing in the event of user authorization failure. This field is required if authorization is not disabled.	string

Example

```
# basic authentication with LDAP registry

- user-security:
    version: 2.0.0
    title: user-security
    factor-id: default
    extract-identity-method: basic
    ei-stop-on-error: true
    user-auth-method: ldap
    ldap-registry: corporate-ldap
    au-stop-on-error: true
    user-az-method: authenticated
    az-stop-on-error: true
```

API Gateway

validate - DataPower API Gateway

Use the validate policy to validate the payload in an assembly flow against a schema.

Gateway support

Note: This page describes the validate policy implementation in the DataPower® API Gateway. If you are using the DataPower Gateway (v5 compatible), see validate - DataPower Gateway (v5 compatible).
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.1.0 or later) 2.2.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.3.0 (DataPower API Gateway Version 10.0.2.0 or later) 2.4.0 (DataPower API Gateway Version 10.0.4.0 or later) 2.5.0 (DataPower API Gateway Version 10.5.0.0 or later) 2.6.0 (DataPower API Gateway Version 10.5.0.2 or later 2.7.0 (DataPower API Gateway Version 10.5.0.3 or later

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Validate - DataPower API Gateway.

About

The validate policy has the following format:

```
- validate:
    version: version
    title: title
    description: description
    validate-against: validation_mechanism
          .
          .
          .
       properties_specific_to_the specified_validation_mechanism
          .
          .
          .
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

Note: With the DataPower API Gateway, the input to the validate policy must be parsed data. One way to produce parsed data is to use a **parse** policy before a validate policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table describes the policy properties:

Table 2. validate policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
input	No	Identifies a variable in the API context. The content of the **body** field of the variable, which is represented by ***variable_name*.body**, is the input data to validate. By default, the variable name is **message**.	string
output	No	Specifies the name of a variable in the API context. • If the validation passes, the body field of the output variable, which is represented by ***variable_name*.body**, stores the output of the assembly validate action. • If the schema to validate is a JSON schema, the validation also adds any default values that are missing from the payload. • If the validation fails, no output is stored. • If an output variable is not specified, the results of the assembly validate action are not saved.	string
validate-against	Yes	Specifies the schema to be used for validating the payload. Valid values: • **definition**: A previously defined schema will be used to validate the payload that is returned from other invoke actions or tasks in the assembly flow. In addition, supply a **definition** property to specify the required schema. • **url**: the schema is identified by a URL location. In addition, supply the following properties: ◦ **json-schema** field: the URL of the JSON schema to be used for validating a JSON payload. ◦ **xml-validation-mode**, specify one of the following values to define how an XML payload is validated: ▪ **xsd**: use an XML schema; in addition, supply an **xml-schema** property that specifies the URL of the XML schema. ▪ **wsdl**: use a WSDL schema; in addition, supply an **xml-schema** property that specifies the URL of the WSDL schema to be used for validating a SOAP payload.. ▪ **soap-body**: validate the body of a SOAP message against the XML schema only. Note: The following limitations apply to schemas used for JSON validation, which include JSON schemas and OpenAPI documents that are used as schemas for validation. Exceeding these limits can impact compilation performance and is not supported. ▪ Maximum of 6,500 lines. Each key and each item in an array count as a line. ▪ Maximum recursion depth of 250. ▪ Maximum of 3,000 items in **enum** lists. • **wsdl**: (use with a SOAP service based API only) use the XML schema in the WSDL file associated with the API operation or the API path. • **body-param**: validate the request input against the schema definition that is specified in the **schema** property for the request parameter for this operation. • **response-param**: validate the response to be returned to the client application, against the schema definition that is specified in the **schema** property for the response parameter for this operation. • **graphql**: (use with a GraphQL proxy API only) validate the payload against the GraphQL schema that has been imported into the GraphQL proxy API. In addition, either the GraphQL query or response, depending on the input, is analyzed against the GraphQL schema to calculate the cost, and the result is placed in the API context. For more information on GraphQL proxy APIs, see Creating a GraphQL proxy API and GraphQL context variables. .	string
xslt-version	No	The XSLT processor version. The default value is XSLT10.	string
strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean

Property	Required	Description	Data type
profile	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
debug	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
stream	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
try-stream	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
minimum-escaping	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
stack-size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
wsi-validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn. Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string
wsdl-validate-body	No	The validation behavior for the **`soap:Body`**. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-validate-headers	No	The validation behavior for the **`soap:Header`**. The default setting is Lax. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-validate-faults	No	Specifies the validation behavior for the fault detail. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
wsdl-wrapped-faults	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
allow-soap-enc-array	No	Whether to allow the schema to accept most uses of elements with **`xsi:type='SOAP-ENC:Array'`** consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the **`xsi:type`** attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, **`xsi:type`** is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from **`SOAP-ENC:Array`**. The opposite is the normal allowable case. By default, elements with **`xsi:type='SOAP-ENC:Array'`** are not accepted.	boolean
validate-soap-enc-array	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean

Property	Required	Description	Data type
wildcards-ignore-xsi-type	No	Whether `xs:any` elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an `xsi:type` attribute on it. This option ignores those `xsi:type` attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, `xsi:type` attributes are not ignored.	boolean
wsdl-strict-soap-version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
xacml-debug	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
allow-xop-include	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an `xop:Include` reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled. • When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as `xs:base64Binary` or `xs:string`, but the message contains an `xop:Include` element instead. • When enabled, an `xop:Include` element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The `xop:Include` element itself will be validated according to the built-in schema in `store:///schemas/xop.xsd`.	boolean

You can also apply a validate policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see Validate - DataPower API Gateway in the built-in policies section.

Example 1

```
- validate:
    version: 2.0.0
    title: 'validate, response parameter schema'
    validate-against: response-param
```

Example 2

```
- validate:
    version: 2.0.0
    title: 'validate, predefined schema'
    validate-against: definition
    definition: '#/definitions/RouteOutput'
```

Example 3

```
- validate:
    version: 2.0.0
    title: 'validate, JSON and XML schema URLs'
    validate-against: url
    json-schema: 'https://my.json.schema'
    xml-validation-mode: xsd
    xml-schema: 'https://my.xml.schema'
```

DataPower Gateway (v5 compatible)

validate - DataPower Gateway (v5 compatible)

Use the validate policy to validate the payload in an assembly flow against a schema.

Gateway support

Note: This page describes the validate policy implementation in the DataPower® Gateway (v5 compatible). If you are using the DataPower API Gateway, see validate - DataPower API Gateway.
For information on the different types of gateway, see API Connect gateway types.

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower Gateway (v5 compatible)	1.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Validate - DataPower Gateway (v5 compatible).

Restriction:

- The schema that represents the XML can reference only one XML namespace.

- The schema cannot reference polymorphic XML elements.
- The validation works on the `message.body` variable and not any other output/context variable. If the invoke policy contains a configured response object variable, then `message.body` is not set, and validate is not able to act.
- If you use the `multipleOf` keyword in a schema definition for the API then, due to rounding behavior, the specified value must satisfy the following conditions, otherwise the validation fails when the API is called:
  - The value must not be less than `0.00000099999999999999848869`.
  - If the value is greater than 1, the amount before the decimal point must not be greater than `99999999999999934463`.

About

The validate policy has the following format:

```
- validate:
  version: version
  title: title
  description: description
  definition: swagger_schema_definition_to_be_used
```

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

Properties

The following table describes the policy properties:

Table 2. validate policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	No	A title for the policy.	string
description	No	A policy description.	string
definition	Yes	The schema to be used to validate the payload. Valid values: • `request`: Select this value to validate the request input against the schema definition that is specified in the Type field for the request parameter for this operation. For information about how to create a request parameter, see Configuring an operation. • `response`: Select this value to validate the response to be returned to the client application, against the schema definition that is specified in the Schema field for the response parameter for this operation. For information about how to create a response parameter, see Configuring an operation. • The name of a schema definition, in the following format: `#/definitions/schema_name` The schema must be defined in the `definitions:` section of your OpenAPI file.	string

You can also apply an validate policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see Validate - DataPower Gateway (v5 compatible) in the built-in policies section.

Example 1

```
validate:
  version: 1.0.0
  title: validate the response
  definition: #/definitions/RouteOutput
```

validate-usernametoken

Use the Validate Username Token policy to validate a Web Services Security (WS-Security) UsernameToken in a SOAP payload before allowing access to the protected resource.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0 1.1.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Validate Username Token.

About

The validate-usernametoken policy has the following format:

```
- validate-usernametoken:
    version: version
    title: title
    description: description
```

```
    auth-type: Authentication URL_or_LDAP Registry (policy version 1.0.0 only)
    auth-url: authentication_url_to_use (policy version 1.0.0 only)
    tls-profile: tls_profile_to_use (policy version 1.0.0 only)
    ldap-registry: name_of_the_ldap_user_registry (policy version 1.0.0 only)
    registry: name_of_the_ldap_or_authurl_user_registry (policy version 1.1.0 and later)
    ldap-search-attribute: name_of_the_ldap_user_password_attribute
```

Properties

The following table describes the policy properties:

Table 2. Validate Username Token policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
auth-type (policy version 1.0.0 only)	Yes	The authentication type to use to validate the UsernameToken. Valid values: • **`Authentication URL`**: Specify this value to validate the user credentials against an authentication URL. • **`LDAP registry`**: Specify this value to validate the user credentials against an LDAP user registry. The default value is: **`Authentication URL`**.	string
auth-url (policy version 1.0.0 only)	Yes	The authentication URL to use to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to **`Authentication URL`**.	string
tls-profile (policy version 1.0.0 only)	No	The TLS profile to use for the secure transmission of data to the authentication URL. Note: This property is available only if Authentication type is set to **`Authentication URL`**.	string
ldap-registry (policy version 1.0.0 only)	Yes	The name of the LDAP user registry to validate the UsernameToken user credentials against. Note: This property is required only if Authentication type is set to **`LDAP registry`**.	string
registry (policy version 1.1.0 and later)	Yes	The name of the LDAP or Authentication URL registry to use to validate the UsernameToken.	string
ldap-search-attribute	Yes	The name of the LDAP user password attribute. Note: This property is required only for an LDAP user registry.	string

Examples

The following example shows an LDAP user registry authentication:

```
- validate-usernametoken:
    version: 1.0.0
    title: "validate-usernametoken"
    auth-type: "LDAP Registry"
    ldap-registry: "wstest"
    ldap-search-attribute: "userPassword"
```

The following example shows an Authentication URL definition:

```
- validate-usernametoken:
    version: 1.0.0
    title: "validate-usernametoken"
    auth-type: "Authentication URL"
    auth-url: "https://www.google.com"
    tls-profile: "default-ssl-profile"
```

For more information about how to use a validate-usernametoken security policy, see Validate Username Token in the built-in policies section.

websocket-upgrade

Use the websocket-upgrade policy to process API requests and responses through a WebSocket connection.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see Websocket Upgrade.

The websocket-upgrade policy has the following format:

```
- websocket-upgrade:
  version: version
  title: title
  description: description
  target-url: URL_of_target_API
```

```
tls-profile: TLS_profile_to_be_used
timeout: timeout_value_in_seconds
follow-redirects: redirect_behavior_on_301_error
username: username_if_authentication_required
password: password_if_authentication_required
inject-proxy-headers: are_proxy_headers_sent_to_target_url
decode-request-params: are_request_parameters_decoded
encode-plus-char: are_plus_characters_encoded
header-control:
      .
      .
      .
  headers_to_copy_to_target_url
      .
      .
      .
parameter-control:
      .
      .
      .
  parameters_to_copy_to_target_url
      .
      .
      .
```

Properties

Table 2. websocket-upgrade policy properties

Property	Required	Description	Data type
version	Yes	The policy version number.	string
title	No	The title of the policy.	string
description	No	A description of the policy.	string
target-url	Yes	Specify the URL to be invoked.	string
tls-profile	No	Specifies a TLS profile to use for the secure transmission of data.	string
timeout	No	The time to wait before a reply back from the endpoint (in seconds). The default value is **60**.	integer
follow-redirects	No	Specifies the behavior if the back-end server returns the HTTP status code **301 Moved Permanently**. If this property is set to **true**, the **invoke** policy follows the URL redirection by making a further call to the URL specified in the **Location** header in the response. If this property is set to **false**, the **invoke** saves the **301** status code and the API call is considered to be complete. Note: The **follow-redirect** property is supported only by the DataPower API Gateway. If you are using the DataPower Gateway (v5 compatible), the **invoke** always follows the URL redirection; the **proxy** policy (not supported by the DataPower API Gateway) saves the **301** status code and completes the API call without following the URL redirection.	boolean
username	No	The username to use for HTTP Basic authentication.	string
password	No	The password to use for HTTP Basic authentication.	string
inject-proxy-headers	No	When set to true, the **invoke** policy injects the **X-Forwarded-For**, **X-Forwarded-To**, **X-Forwarded-Host**, and **X-Forwarded-Proto** headers to the request that is sent to the t**target-url**. The default value is **false**.	boolean
decode-request-params	No	When set to true, any request parameters that are referenced by a variable definition on the **target-url** of the invoke policy are URL-decoded. The default value is **false**.	boolean
encode-plus-char	No	When set to true, all "+" characters in the query parameter values of the **target-url** are encoded to %2F. The default value is **false**.	boolean
header-control: **type** **values**	No	Specifies the headers in **message.headers** that you want to copy to the target URL. If the **type** property is set to **blocklist**, the **values** property lists the headers that you don't want to be copied. If the **values** property is empty then all headers are copied. If the **type** property is set to **allowlist**, the **values** property lists the headers that you want to be copied. The items that are listed in the **values** property are in regular expression format. The values are not case-sensitive. The default value of the **header-control** property is **header-control:** **type: blocklist** **values: []** See header-control examples	object
parameter-control: **type** **values**	No	Specifies the parameters in the incoming request that you want to be copied to the target URL. If the **type** property is set to **blocklist**, the **values** property lists the parameters that you don't want to be copied. If the **type** property is set to **allowlist**, the **values** property lists the parameters that you want to be copied. If the **values** property is empty then no parameters are copied. The items that are listed in the **values** property are in regular expression format. The values are not case-sensitive. The default value of the **parameter-control** property is **parameter-control:** **type: allowlist** **values: []** See parameter-control examples	object

Property	Required	Description	Data type
request-assembly: **execute** . . . ***policy assembly*** . . .	No	The request processing assembly. For details on configuring an assembly in the **execute** section, see execute.	object
response-assembly: **execute** . . . ***policy assembly*** . . .	No	The response processing assembly. For details on configuring an assembly in the **execute** section, see execute.	object

Example

```
- websocket-upgrade:
  version: 2.0.0
  title: websocket-upgrade
  timeout: 60
  target-url: 'https://my.websocket.upgrade.com'
  secure-gateway: false
  follow-redirects: true
  inject-proxy-headers: true
  decode-request-params: false
  encode-plus-char: true
  response-assembly:
    execute:
      - parse:
          version: 2.0.0
          title: parse
          parse-settings-reference:
            default: apic-default-parsesettings
          description: Parse the response from the backend server
```

`header-control` examples

```
# copy all headers to the target URL

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: blocklist
      values: []

# copy all headers except X-Client-ID and Content-Type

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: allowlist
      values:
        - ^X-Client-ID$
        - ^Content-Type$

# copy no headers

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: allowlist
      values: []

# copy only the Content-Type header

- invoke:
    target-url: http://myhost/mypath
    header-control:
      type: allowlist
      values:
        - ^Content-Type$
```

`parameter-control` examples

```
# copy no request parameters to the target URL

- invoke:
    target-url: http://myhost/path?storeid=3
    parameter-control:
      type: allowlist
      values: []
```

```
# append the petid parameter to the target URL
# if the incoming request is http://apigw/org/sandbox/petstore/base?petid=100&display=detailed,
# the target URL at runtime will be http://myhost/mypath?storeid=3&petid=100

- invoke:
    target-url: http://myhost/path?storeid=3
    parameter-control:
      type: allowlist
      values:
        - ^petid$
```

xml-to-json

Use the xml-to-json policy to convert the context payload of your API from the extensible markup language (XML) format to JavaScript Object Notation (JSON).

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see XML to JSON.

About

The xml-to-json policy has the following structure:

```
- xml-to-json:
   version: version
   title: Title
   description: Description
```

Note: If you are using the DataPower API Gateway, the input to the xml-to-json policy must be parsed data. One way to produce parsed data is to use a **parse** policy before an xml-to-json policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table describes the policy properties:

Table 2. Policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
API Gateway only input	No	The input message to convert. Specify the name of a variable in the API context. ***variableName*.body**, the message payload, represents the JSON input to convert. The default value of the variable is **message** and **message.body** is the default input.	string
API Gateway only output	No	The output message to store the conversion result. Specify the name of a variable in the API context. ***variableName*.body** represents the result of conversion from JSON format to XML format. When the specified input message is the default message, the default output is **message.body**. Otherwise, when the input message is the variable **my-message-variable**, for example, the default output is **my-message-variable.body**. The variable cannot be any read-only in the API context.	string
API Gateway only conversionType	No	The conversion type that determines the target format of the output. The following options are available: • badgerFish: BadgerFish convention is used to determine the target conversion format of the output. • apicv5: apicv5 convention is used to determine the target conversion format of the output.	string

Example

The following is an example of a xml-to-json policy:

```
- xml-to-json:
   version: 1.0.0
   title: XML to JSON transform
   description: Transforms XML message body to JSON format
```

xslt

Use the xslt policy to apply an XSLT transform to the payload of the API definition.

Gateway support

Table 1. Table showing which gateways support this policy, and the corresponding policy version

Gateway	Policy version
DataPower® Gateway (v5 compatible)	1.0.0
DataPower API Gateway	2.0.0 2.1.0 (DataPower API Gateway Version 10.0.4.0 or later)

This topic describes how to configure the policy in your OpenAPI source; for details on how to configure the policy in the assembly user interface, see XSLT.

About

The xslt policy has the following structure:

```
- xslt:
    version: version
    title: Title
    description: Description
    input: Input_True_False
    source: Transform
```

Note: If you are using the DataPower API Gateway, the input to the xslt policy must be parsed data. One way to produce parsed data is to use a **parse** policy before an xslt policy in your assembly flow, which provides explicit control of the parse action.

Properties

The following table describes the policy properties:

Table 2. xslt policy properties

Property	Required	Description	Data type
version	Yes	The policy version number	string
title	Yes	The title of the policy.	string
description	No	A description of the policy.	string
input	No	Indicates whether this XSLT input document uses the context current payload, or if there is no input. The default value is **false**.	boolean
API Gateway only serialize-output	No	If set to **true**, the output tree that is generated by the XSLT policy is serialized. The content of **message.body** is updated with the serialized binary data rather than the XML tree. The default value is **false**.	boolean
source	Yes	The XSLT transform source to execute.	string
API Gateway only xslt-version	No	The XSLT processor version. The default value is XSLT10.	string
API Gateway only strict	No	Whether to enable strict XSLT error checking. Non-strict operations attempt to recover from certain errors, such as use of undeclared variables, calling undeclared templates, and so forth. By default, strict XSLT error checking is enabled.	boolean
API Gateway only profile	No	Whether to enable stylesheet profiling. This option should not be used in production environments. By default, stylesheet profiling is disabled.	boolean
API Gateway only debug	No	Whether to run the stylesheet, XQuery script, and JSONiq script in debug mode. When a stylesheet, XQuery script, or JSONiq script is run in debug mode, it generates a custom web page instead of displaying its normal output. The web page details exactly what occurred during execution, including the values of variables and where particular pieces of the output came from. This option should not be used in production environments. By default, debug mode is disabled.	boolean
API Gateway only stream	No	Whether the stylesheet must be run in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, an error is generated and the input is not processed. By default, streaming mode is disabled.	boolean
API Gateway only try-stream	No	Whether to attempt to run the stylesheet in streaming mode. Transformation of the document begins before the input is fully parsed. Not all stylesheets can be streamed. If the stylesheet cannot be streamed, a warning is generated during compilation and the stylesheet is read in the entire input as normal at execution time. By default, attempting to run the stylesheet in streaming mode is disabled.	boolean
API Gateway only minimum-escaping	No	Whether to escape output produced from the stylesheet during processing. Minimal escaping is particularly useful when handling non-English character sets. By default, minimum escaping is disabled.	boolean
API Gateway only stack-size	No	The maximum number of bytes that the stack is allowed to use while executing a stylesheet or other compiled content. This setting is used to block infinite recursion. The minimum value is 10 kilobytes, or 10,240 bytes. The maximum value is 100 megabytes, or 104,857,600 bytes. The default value is 1 megabyte, or 1,048,576 bytes.	integer
API Gateway only wsi-validation	No	The validation behavior to apply to WSDL files that are checked for conformance to section 5 of WS-I Basic Profile (version 1.0, April 2004). The default setting is Warn. Ignore Disables conformance checking. Warn Logs warnings for violations. Fail Forces conformance. Fails if the file contains any violation.	string

Property	Required	Description	Data type
API Gateway only wsdl-validate-body	No	The validation behavior for the **soap:Body**. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
API Gateway only wsdl-validate-headers	No	The validation behavior for the **soap:Header**. The default setting is Lax. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
API Gateway only wsdl-validate-faults	No	Specifies the validation behavior for the fault detail. The default setting is Strict. Strict Validates this part of the message against the WSDL definition. Allows only messages that contain this part and that match the WSDL definition. Lax Validates this part of the message against the WSDL definition. If the message contains this part and the WSDL definition does not define this part, allow the message. If the message contains this part and the WSDL definition defines this part, allow the message only when there is a match. Skip Disables validation of this part of the message.	string
API Gateway only wsdl-wrapped-faults	No	Whether to require compatibility with RPC-style wrappers. By default, RPC-style wrappers are not required.	boolean
API Gateway only allow-soap-enc-array	No	Whether to allow the schema to accept most uses of elements with **xsi:type='SOAP-ENC:Array'** consistent with SOAP 1.1 Section 5, even when these attributes violate the XML Schema specification. Normally, the **xsi:type** attribute must name a type equal to or derived from the actual type of the element. For schemas compiled with this option, **xsi:type** is accepted specifically for the SOAP 1.1 Encoding 'Array' complex type if the element type is derived from **SOAP-ENC:Array**. The opposite is the normal allowable case. By default, elements with **xsi:type='SOAP-ENC:Array'** are not accepted.	boolean
API Gateway only validate-soap-enc-array	No	Whether to perform extra schema validation following the encoding rules in SOAP 1.1 Section 5. When enabled, members of SOAP arrays are validated, attributes such as @id and @href are allowed even if they are not allowed by the schema, and @href values are checked to ensure that they have a corresponding @id element. By default, the extra validation is not performed.	boolean
API Gateway only wildcards-ignore-xsi-type	No	Whether **xs:any** elements in the schema validate only child elements by name. The XML Schema specification requires that, if a wildcard matches an element but that element does not have an element declaration, the element is instead validated according to an **xsi:type** attribute on it. This option ignores those **xsi:type** attributes. It should be used for cases such as SOAP envelope validation where a further validation step will validate the contents matching the wildcard, possibly using the SOAP 1.1 encoding rules. By default, **xsi:type** attributes are not ignored.	boolean
API Gateway only wsdl-strict-soap-version	No	Whether to strictly follow the SOAP binding in the WSDL. When enabled, only messages bound to SOAP 1.2 appear in SOAP 1.2 envelopes and only messages bound to SOAP 1.1 appear in SOAP 1.1 envelopes. By default, strict SOAP binding is disabled.	boolean
API Gateway only xacml-debug	No	Whether to compile XACML policies with debug information. Note that the XACML debugging messages are also controlled by the log event in the XACML category. Use the debug log level to view the full XACML debugging messages. By default, XACML policies are not compiled with debug information.	boolean
API Gateway only allow-xop-include	No	Specifies whether the schema or WSDL document accepts messages where base64-encoded binary content was optimized according to the MTOM/XOP specifications. XOP binary-optimization replaces base64-encoded binary data with an **xop:Include** reference element that references the unencoded binary data located in an attachment. By default, MTOM/XOP optimized messages are disabled. • When disabled, such optimized messages are rejected by validation of the optimized form. Rejection occurs because the schema specifies a simple type that accepts base64-encoded data, such as **xs:base64Binary** or **xs:string**, but the message contains an **xop:Include** element instead. • When enabled, an **xop:Include** element can optionally appear in place of content for any XML Schema simple type that validates base64-encoded binary data. The **xop:Include** element itself will be validated according to the built-in schema in **store:///schemas/xop.xsd**.	boolean

You can also apply an xslt policy by using the API Designer assembly editor to add a built-in policy to the API. For more information, see XSLT in the built-in policies section.

For examples of the OpenAPI definitions of xslt policies, see XSLT policy examples in the built-in policies section.

For more examples of how to use XSLT to access and modify properties and context, see Implementation code examples and, if you are using the DataPower API Gateway, Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

Related concepts

- Variable references in API Connect

Related reference

- XSLT
- API Connect context variables

catch

Use the catch extension to catch errors that occur during an API call.

The catch extension takes the following form:

```
catch:
   - errors:
      - Error_1
      - Error_2
   - execute:
      assembly_1
   - errors:
      - Error_3
   - execute:
      assembly_2
   - default:
      assembly_3
```

The following table shows the properties of the **catch** extension:

Table 1. The properties of the catch extension

Property	Required	Description	Type
errors	Yes	The errors for which the catch will activate. For a list of errors, see Error cases supported by assembly catches.	Array (String)
execute	Yes	The section of the assembly that will execute when the catch is activated.	Array (Object)
default	No	The section of the assembly that will execute when an error does not trigger other catches. It behaves in the same manner and has the same structure as **execute**	Array (Object)

The following is an example of a **catch** extension:

```
    catch:
      - errors:
          - ConnectionError
          - JavaScriptError
        execute:
          - activity-log:
              title: activity-log
              content: activity
              error-content: payload
      - default:
          - activity-log:
              title: activity-log
              content: activity
              error-content: payload
```

properties

Use the **properties** extension to define properties for referencing in an API.

The **properties** extension has the following structure:

```
properties:
   property_1:
      value: default_value_1
      description: description_1
      encoded: encoded_boolean_1
   property_2
      value: default_value_2
      description: description_2
      encoded: encoded_boolean_2
```

The following table lists the fields found in the **properties** extension:

Table 1. The properties extension

Property	Required	Description	Data type
property	Yes	The name of the property. It is used when referencing the property. Note that this is the field name, not the contents of the field.	Object
value	No	The default value that the property takes. It can be empty.	String
description	No	The description of the property. It can be empty.	String

Property	Required	Description	Data type
`encoded`	No	Specifies whether to encode the value of this property.	Boolean

The following example shows a sample `properties` field:

```
properties:
  ID:
    value: 1234
    description: An ID to be used for validating.
    encoded: false
```

catalogs

Use the `catalogs` extension to assign catalog-specific values to properties defined in the `properties` section.

The `catalogs` section has the following structure:

```
catalogs:
   catalog_name_1:
      properties:
         property_name_1: value_1
         property_name_2: value_2
   catalog_name_2:
      properties:
         property_name_1: value_3
```

The following table lists the fields found in the `catalogs` extension:

Table 1. Catalogs properties

Property	Required	Description	Data type
`catalog_name`	Yes	The name of the catalog. It must match a catalog in API Manager. Note that this is the field name, not the contents of the field.	Object
`properties`	Yes	This field contains any properties that are to be set.	Object
`property_name`	Yes	The field name is the property to be set and must match a property defined in the `properties` extension. It contains the value of the property for the catalog to which the field belongs. If, in the `properties` extension, it has been specified as encoded, this will be encoded when saved or staged by IBM® API Connect.	String

The following is an example of a `catalogs` extension:

```
catalogs:
  Sandbox:
    properties:
      ID: 5678
```

API Gateway only

activity-log

If you're using the DataPower® API Gateway, you can use the `activity-log` extension to configure your logging preferences for the API activity that is stored in analytics. The preferences that you specify will override the default settings for collecting and storing details of the API activity.

An API event record exists for each API execution event in the Gateway server. By default, the content type that is collected and stored in API event records is `activity` for when API execution completes successfully, and `payload` for when API execution completes with an error code. When you compose your API definition, you can change the type of content to log in these API event records. For more information about API event records, see API event record field reference.

Note that if you're using the DataPower Gateway (v5 compatible), you can configure your logging preferences by using an `activity-log` policy in your API assembly. For more information, see activity-log.

The `activity-log` extension takes the following form:

```
activity-log:
  success-content: activity_to_log_if_call_successful
  error-content: activity_to_log_if_call_unsuccessful
  enabled: is_activity_logging_enabled
```

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
  success-content: activity_to_log_if_call_successful
  error-content: activity_to_log_if_call_unsuccessful
  enabled: is_activity_logging_enabled
buffering: true
```

The following table shows the properties of the `activity-log` extension:

Table 1. The properties of the activity-log extension

Property	Required	Description	Type

Property	Required	Description	Type
`success-content`	No	Defines the type of content to be logged when the operation is successful. Valid values: • **`none`**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **`activity`**: Logs invocation only (only the resource URI is recorded). • **`header`**: Logs activity and header. • **`payload`**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **`activity`**.	String
`error-content`	No	Indicates what content to log when an error occurs. Valid values: • **`none`**: Indicates that no logging occurs. Restriction: This option disables notifications for application developers who use your Developer Portal. • **`activity`**: Logs invocation only (only the resource URI is recorded). • **`header`**: Logs activity and header. • **`payload`**: Logs activity, header, and payload (the original request, if any, and the final response). The default value is **`payload`**.	String
`enabled`	No	Indicates whether activity logging is enabled or disabled.	Boolean

Example 1

Override the default activity logging settings:

```
activity-log:
    success-content: none
    error-content: header
    enabled: true
```

Example 2

Turn off activity logging:

```
activity-log:
    enabled: false
```

Specifying a gateway type for an API definition

An API definition is specific to one or other of the gateway types, DataPower® API Gateway or DataPower Gateway (v5 compatible). API definitions must specify which type of gateway the API uses.

Before you begin

IBM® API Connect supports two gateway types: DataPower Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see API Connect gateway types.

About this task

You must specify which type of gateway each API uses. APIs can use only one type of gateway.

When you modify your API definitions to use a specific gateway type, you must ensure that each policy and policy version in the API are supported by the gateway type. DataPower Gateway (v5 compatible) and DataPower API Gateway each support policies that the other gateway type does not. In some cases, the same policy is supported by both gateway types, but with a different version number.

For example, DataPower Gateway (v5 compatible) supports version **1.0.0** of the **invoke** policy, but DataPower API Gateway requires version **2.0.0**.

For information on policies, see execute.

For information on policy versions, see the documentation for each individual policy. For example, to review the **invoke** policy, see invoke.

Procedure

1. Edit the API definition YAML file and add the gateway type. For example:
   - DataPower Gateway (v5 compatible):

     ```
     gateway: datapower-gateway
     ```

   - DataPower API Gateway

     ```
     gateway: datapower-api-gateway
     ```

Note: If your API YAML file includes the `gateway:` property, the `enforced:` property must be set to `true`

2. Ensure that the policies, including policy versions, in your APIs are supported by the gateway type. Review the list of policies on the execute page. If necessary, review the individual policy pages to determine the supported version number.
3. Ensure that any Product that uses this API is configured to use the same gateway type. See Specifying a gateway type for your Product.

Related information

- DataPower API Gateway porting notes

Referring to an extension in an API definition

To refer to an OpenAPI extension in your API definition YAML (or JSON) file, add an **`extensions`** key under **`x-ibm-configuration`**.

Before you begin

Add the OpenAPI extension to any Catalogs that the API is to be published to, by importing its definition file; for details, see Working with OpenAPI extensions. If the target Catalog does not contain the OpenAPI extension then attempts to publish the API to it will fail.

About this task

The following example shows an extension definition file that defines a bank branch, in YAML format:

```
extension: '1.0.0'

info:
   title: Banking services
   name: banking
   version: 1.0.0
   description: Banking extensions
   contact:
     name: IBM API Connect
     url: https://apiconnect.ibm.com/
     email: myname@ibm.com

portal-visible: true

properties:
  title: "Branch"
  type: "object"
  properties:
    Branch type:
      type: "string"
      enum:
        - "ATM"
        - "Walk in"
    location:
      type: "object"
      title: "Location"
      properties:
        city:
          type: "string"
          default: "San Francisco"
        state:
          type: "string"
          default: "CA"
        citystate:
          type: "string"
          description: "This is generated automatically from the previous two fields"
          template: "{{city}}, {{state}}"
          watch:
            city: "location.city"
            state: "location.state"
  required:
    - Branch type
```

Procedure

1. Edit the API definition YAML file and add the reference. For example:

   ```
   x-ibm-configuration:
         .
         .
         .
     extensions:
       banking: 1.0.0
   ```

2. In your API definition file, add a property whose name must begin with **`x-`** and references the name of the extension you created. Specify the values of the properties that were defined in the OpenAPI extension YAML file.
   The following example uses the banking extension:

   ```
         .
         .
         .
   ```

```
x-banking:
  Branch type: ATM
  location:
    city: San Francisco
    state: CA
    citystate: 'San Francisco, CA'
```

Using $ref to reuse code fragments in your OpenAPI files

If you deploy an API to an IBM® API Connect Management server by using the developer toolkit command line, you can use the **$ref** field in your OpenAPI YAML and JSON API definition files to reference a fragment of OpenAPI code that is defined in a separate file. When API Connect processes the source API definition file, the **$ref** field is replaced with the contents of the target file.

Use the following syntax in your source YAML file:

```
$ref: path_to_file_containing_code_fragment
```

Use the following syntax in your source JSON file:

```
{
$ref: path_to_file_containing_code_fragment
}
```

For example:

```
$ref: code_fragments/my_fragment.yaml
```

```
{
  "$ref": "code_fragments/my_fragment.json"
}
```

The replacement of the **$ref** field with the target code fragment occurs when you perform any of the following actions on the API defined by the source API definition file:

- Stage or publish an API to an API Connect Management server by using the **apic publish** command. For more information, see Publishing APIs and applications.
- Validate the API definition YAML file by using the **apic validate** command. For more information, see Validating the YAML or JSON definition of an API or Product.

Important: You cannot insert a $ref field at the root level of your OpenAPI file.

Example 1

A source YAML file contains the following OpenAPI code:

```
swagger: '2.0'
info:
  version: 1.0.0
  title: Branches
  x-ibm-name: Branches
  description: Provides operations relating to BankA branch information.
basePath: /branches
paths:
  /details:
    get:
      responses:
        - $ref: code_fragments/paths.yaml#/details/get/responses
        .
        .
        .
```

The file paths.yaml contains the following OpenAPI code fragment:

```
details:
  get:
    responses:
      '200':
        description: 200 OK defined in $ref file
        schema:
          $ref: '#/definitions/branch'
    summary: Branch details
    description: Retrieve details of the current branches of BankA.
```

When API Connect processes the source YAML file, the **$ref** field is replaced with the target code fragment, yielding the following OpenAPI code:

```
swagger: '2.0'
info:
  version: 1.0.0
  title: Branches
  x-ibm-name: Branches
  description: Provides operations relating to BankA branch information.
basePath: /branches
paths:
  /details:
    get:
      responses:
        '200':
          description: 200 OK
          schema:
            $ref: '#/definitions/branch'
      summary: Branch details
```

```
      description: Retrieve details of the current branches of BankA.
       .
       .
       .
```

Example 2

Example of a reference to an external file that contains just the parameter details:

```
openapi: 3.0.0

info:
  version: 3.0.1
  title: BeamUp API
  x-ibm-name: BeamUpAPI
servers:
  - url: /
paths:
  /customers:
    get:
      parameters:
        - $ref: tests/fixtures/api/referenceFile.yaml
      responses:
        200:
          description: An array of customers
          content:
            application/json:
              schema:
                type: object
                properties:
                  customerID:
                    type: string
                  customerName:
                    type: string
.
.
.
```

The file referenceFile.yaml contains the following OpenAPI code fragment:

```
name: page
in: query
description: page of results to return
required: true
schema:
  type: integer
```

Example of a reference to an external file that contains the parameters inside a path code fragment:

```
openapi: 3.0.0

info:
  version: 3.0.1
  title: BeamUp API
  x-ibm-name: BeamUpAPI
servers:
  - url: /
paths:
  /customers:
    get:
      parameters:
        - $ref: tests/fixtures/api/referenceFileWithPath.yaml#/customers/get/parameters
      responses:
        200:
          description: An array of customers
          content:
            application/json:
              schema:
                type: object
                properties:
                  customerID:
                    type: string
                  customerName:
                    type: string
```

The file referenceFileWithPath.yaml contains the following OpenAPI code fragment:

```
customers:
  get:
    parameters:
      name: page
      in: query
      description: page of results to return
      required: true
      schema:
        type: integer
```

Creating a Product definition file

You define a Product by creating a Product definition file.

About this task

You can create a Product definition file in either of the following ways:

- Create the file in an editor of your choice.
- Create a Product definition file by using the **`apic create:product`** command and then modify it. You can base your Product definition file on the default Product template or on your own custom template.

This topic describes the syntax that is needed for both options and also the structure needed when the file is created in the editor. For details on what you can include in the product definition file, see Product definition schema.
You can create a Product in the CLI by running **`apic create:product`** and supplying additional arguments on the command line. For example, you can enter the following text in a single command:

```
apic create:product --name product_name --title product_title --filename product_file_name.yaml
--apis "filename_of_api1.yaml filename_of_api2.yaml"
```

where:

- *product_name* is the name for your new Product.
- *product_title* is the title of your new Product.
- *product_file_name.yaml* is the name of the yaml file that is created for your new Product.
- *filename_of_api1.yaml* is the file name of one of the APIs that is used within the new Product.
- *filename_of_api2.yaml* is the file name of one of the APIs that is used within the new Product.

Another option is to create a Product interactively in the command line by running **`apic create:product`** and following the prompts.
You can see further details and available options for the **`apic create:product`** command by entering the command:

```
apic create:product --help
```

You can also create a Product from a custom Handlebars template by using the following command:

```
apic create:product --template template_filename --title product_title
```

where *template_filename* is the name of the Handlebars template to use, and *product_title* is the title of your Product. A Product template file must have a .hbs file name extension. You can create a template from scratch, or start with the example (default) Product template provided in API and Product definition template examples.
A Product definition file contains the following sections:

- The specification version
- An information section
- A visibility section
- An APIs section
- A Plans section

In this topic, YAML is used, but the instructions can be adapted to use JSON. Both .yaml and .yml file extensions are supported, but the use of the .yaml file extension is recommended by yaml.org.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

Procedure

Structure your Product definition file by completing the following steps:

1. Set the specification version by adding the following line to the beginning of the file: `product: 1.0.0`.
   This line specifies the template version, and is always **`product: 1.0.0`**.
   Note: The specification version is distinct from your Product version. The specification version refers to this YAML file, while the Product version is by your discretion.
2. Include an information section with details about the Product, as described in Completing the information section of your Product description.
3. Include a visibility section that specifies who can view and subscribe to the Product, as described in Specifying the visibility of your Product.
4. Include an APIs section that references the APIs to be included in the Product, as described in Referencing the APIs for your Product.
5. Include a Plans section that described the Plans you want include in your Product, as described in Describing Plans in your Product.

Results

You have completed a YAML representation of your Product. A complete example with full indentation can be found in An example YAML representation of a Product.

You can create multilingual API and Product documentation by using an **`x-ibm-languages`** extension directly in the OpenAPI definition. For more information, see Using x-ibm-languages to create multilingual API and Product documentation.

- **Product definition schema**
  The product definition schema defines the properties (and their data types) that you can include in a product definition file.
- **Converting a Product YAML file to use DataPower API Gateway**
  IBM API Connect now supports DataPower® API Gateway. You can optionally specify this gateway type in your Product YAML file.

Related information

- Changing the availability of a Product

Product definition schema

The product definition schema defines the properties (and their data types) that you can include in a product definition file.

```
{
  "id": "https://api.ibm.com/product/schema.json#",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "A schema to represent the IBM specific API configuration",
  "type": "object",
  "additionalProperties": false,
  "required": [
    "product",
    "info"
  ],
  "patternProperties": {
    "^x-": {}
  },
  "properties": {
    "product": {
      "type": "string",
      "enum": [
        "1.0.0"
      ]
    },
    "info": {
      "$ref": "#/definitions/InfoSection"
    },
    "visibility": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "view",
        "subscribe"
      ],
      "patternProperties": {
        "^x-": {}
      },
      "properties": {
        "view": {
          "type": "object",
          "additionalProperties": false,
          "required": [
            "type"
          ],
          "properties": {
            "type": {
              "enum": [
                "public",
                "authenticated",
                "custom"
              ]
            },
            "orgs": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "tags": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "enabled": {
              "type": "boolean"
            }
          }
        },
        "subscribe": {
          "type": "object",
          "additionalProperties": false,
          "required": [
            "type"
          ],
          "properties": {
            "type": {
              "enum": [
                "authenticated",
                "custom"
              ]
            },
            "orgs": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "tags": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "enabled": {
              "type": "boolean"
            }
```

```
        }
      }
    }
  },
  "properties": {
    "type": "object",
    "propertyNames": {
      "pattern": "^([^ ])|([^ ].*[^ ])$"
    },
    "additionalProperties": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "value": {
          "type": "string"
        },
        "description": {
          "type": "string"
        },
        "encoded": {
          "type": "boolean"
        }
      }
    }
  },
  "catalogs": {
    "type": "object",
    "propertyNames": {
      "pattern": "^([^ ])|([^ ].*[^ ])$"
    },
    "additionalProperties": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "properties": {
          "type": "object",
          "additionalProperties": {
            "type": "string"
          }
        }
      }
    }
  },
  "gateways": {
    "type": "array",
    "minItems": 1,
    "items": {
      "enum": [
        "micro-gateway",
        "datapower-gateway",
        "datapower-api-gateway",
        "event-gateway"
      ]
    }
  },
  "apis": {
    "type": "object",
    "propertyNames": {
      "pattern": "^([^ ])|([^ ].*[^ ])$"
    },
    "additionalProperties": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "$ref": {
          "type": "string"
        },
        "name": {
          "type": "string"
        }
      }
    }
  },
  "oauthProviders": {
    "type": "array",
    "propertyNames": {
      "pattern": "^([^ ])|([^ ].*[^ ])$"
    },
    "items": {
      "type": "string"
    }
  },
  "billings": {
    "type": "object",
    "propertyNames": {
      "pattern": "^([^ ])|([^ ].*[^ ])$"
    },
    "additionalProperties": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
       "name": {
          "type": "string"
        }
      }
    }
```

```
    },
    "plans": {
      "type": "object",
      "propertyNames": {
        "pattern": "^([^ ])|([^ ].*[^ ])$"
      },
      "additionalProperties": {
        "type": "object",
        "additionalProperties": false,
        "patternProperties": {
          "^x-": {}
        },
        "required": [
          "title"
        ],
        "properties": {
          "title": {
            "type": "string"
          },
          "description": {
            "type": "string"
          },
          "approval": {
            "type": "boolean"
          },
          "rate-limit": {
            "$ref": "#/definitions/RateLimitSection"
          },
          "rate-limits": {
            "$ref": "#/definitions/RateLimitsSection"
          },
          "burst-limits": {
            "$ref": "#/definitions/BurstLimitsSection"
          },
          "assembly-rate-limits": {
            "$ref": "#/definitions/AssemblyRateLimitsSection"
          },
          "assembly-burst-limits": {
            "$ref": "#/definitions/AssemblyBurstLimitsSection"
          },
          "assembly-count-limits": {
            "$ref": "#/definitions/AssemblyCountLimitsSection"
          },
          "billing": {
            "$ref": "#/definitions/BillingSection"
          },
          "graphql": {
            "$ref": "#/definitions/GraphqlSection"
          },
          "apis": {
            "type": "object",
            "propertyNames": {
              "pattern": "^([^ ])|([^ ].*[^ ])$"
            },
            "additionalProperties": {
              "type": "object",
              "additionalProperties": false,
              "properties": {
                "operations": {
                  "type": "array",
                  "items": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {
                      "operationId": {
                        "type": "string"
                      },
                      "path": {
                        "type": "string"
                      },
                      "operation": {
                        "type": "string"
                      },
                      "rate-limit": {
                        "$ref": "#/definitions/RateLimitSection"
                      },
                      "rate-limits": {
                        "$ref": "#/definitions/RateLimitsSection"
                      }
                    }
                  }
                },
                "graphql-schema-options": {
                  "$ref": "#/definitions/GraphQLSchemaOptions"
                }
              }
            }
          }
        }
      }
    }
  },
  "definitions": {
    "GraphQLSchemaOptions": {
      "type": "object",
      "additionalProperties": false,
```

```
    "required": [
      "visibility-list"
    ],
    "properties": {
      "visibility-list": {
        "type": "object",
        "additionalProperties": false,
        "required": [
          "type",
          "values"
        ],
        "properties": {
          "type": {
            "type": "string",
            "enum": [
              "remove"
            ]
          },
          "values": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/NonEmptyString"
            }
          }
        }
      }
    }
  },
  "InfoSection": {
    "type": "object",
    "additionalProperties": false,
    "required": [
      "title",
      "version"
    ],
    "patternProperties": {
      "^x-": {}
    },
    "properties": {
      "name": {
        "type": "string",
        "pattern": "^([a-zA-Z0-9._]+(-)*)*([a-zA-Z0-9])$"
      },
      "version": {
        "type": "string"
      },
      "title": {
        "type": "string"
      },
      "description": {
        "type": "string"
      },
      "termsOfService": {
        "type": "string"
      },
      "contact": {
        "type": "object",
        "additionalProperties": false,
        "patternProperties": {
          "^x-": {}
        },
        "properties": {
          "name": {
            "type": "string"
          },
          "url": {
            "type": "string",
            "format": "uri"
          },
          "email": {
            "type": "string",
            "format": "email"
          }
        }
      },
      "license": {
        "type": "object",
        "additionalProperties": false,
        "patternProperties": {
          "^x-": {}
        },
        "required": [
          "name"
        ],
        "properties": {
          "name": {
            "type": "string"
          },
          "url": {
            "type": "string",
            "format": "uri"
          }
        }
      },
      "summary": {
        "type": "string",
```

```
        "description": "Short description of the API."
      },
      "categories": {
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "RateLimitSection": {
    "type": "object",
    "additionalProperties": false,
    "required": [
      "value"
    ],
    "patternProperties": {
      "^x-": {}
    },
    "properties": {
      "value": {
        "$ref": "#/definitions/RateLimitValue"
      },
      "hard-limit": {
        "$ref": "#/definitions/RateLimitHardLimit"
      }
    }
  },
  "AssemblyRateLimitSection": {
    "type": "array",
    "minItems": 1,
    "items": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "value",
        "hard-limit",
        "cache-only",
        "is-client",
        "use-api-name",
        "use-app-id",
        "use-client-id"
      ],
      "patternProperties": {
        "^x-": {}
      },
      "properties": {
        "value": {
          "$ref": "#/definitions/RateLimitValue"
        },
        "hard-limit": {
          "$ref": "#/definitions/RateLimitHardLimit"
        },
        "cache-only": {
          "type": "boolean"
        },
        "is-client": {
          "type": "boolean"
        },
        "use-api-name": {
          "type": "boolean"
        },
        "use-app-id": {
          "type": "boolean"
        },
        "use-client-id": {
          "type": "boolean"
        },
        "dynamic-value": {
          "type": "string"
        },
        "weight": {
          "$ref": "#/definitions/NonEmptyString"
        }
      }
    }
  },
  "AssemblyBurstLimitSection": {
    "type": "array",
    "minItems": 1,
    "items": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "value",
        "cache-only",
        "is-client",
        "use-api-name",
        "use-app-id",
        "use-client-id"
      ],
      "patternProperties": {
        "^x-": {}
      },
      "properties": {
        "value": {
```

```
          "$ref": "#/definitions/RateLimitValue"
        },
        "cache-only": {
          "type": "boolean"
        },
        "is-client": {
          "type": "boolean"
        },
        "use-api-name": {
          "type": "boolean"
        },
        "use-app-id": {
          "type": "boolean"
        },
        "use-client-id": {
          "type": "boolean"
        },
        "dynamic-value": {
          "type": "string"
        },
        "weight": {
          "$ref": "#/definitions/NonEmptyString"
        }
      }
    }
  },
  "NonEmptyString": {
    "type": "string",
    "pattern": ".+"
  },
  "AssemblyCountLimitSection": {
    "type": "array",
    "minItems": 1,
    "maxItems": 1,
    "items": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "value",
        "hard-limit",
        "cache-only",
        "is-client",
        "use-api-name",
        "use-app-id",
        "use-client-id"
      ],
      "patternProperties": {
        "^x-": {}
      },
      "properties": {
        "value": {
          "oneOf": [
            {
              "type": "integer"
            },
            {
              "type": "string",
              "enum": [
                "unlimited"
              ]
            }
          ]
        },
        "hard-limit": {
          "$ref": "#/definitions/RateLimitHardLimit"
        },
        "cache-only": {
          "type": "boolean"
        },
        "is-client": {
          "type": "boolean"
        },
        "use-api-name": {
          "type": "boolean"
        },
        "use-app-id": {
          "type": "boolean"
        },
        "use-client-id": {
          "type": "boolean"
        },
        "dynamic-value": {
          "type": "string"
        },
        "weight": {
          "$ref": "#/definitions/NonEmptyString"
        },
        "auto-decrement": {
          "type": "boolean"
        }
      }
    }
  },
  "RateLimitValue": {
    "type": "string",
    "pattern": "^(([0-9]|[1-9][0-9]{1,8}|[1-3][0-9]{9}|4[01][0-9]{8}|42[0-8][0-9]{7}|429[0-3][0-9]{6}|4294[0-8][0-9]
```

```
{5}|42949[0-5][0-9]{4}|429496[0-6][0-9]{3}|4294967[01][0-9]{2}|42949672[0-8][0-9]|429496729[0-5])\\/([0-9]|[1-9][0-9]{1,3}|[1-
5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-2][0-9]|6553[0-5])?(second|minute|hour|day|week)|unlimited)$"
    },
    "RateLimitHardLimit": {
      "type": "boolean",
      "default": false
    },
    "RateLimitsSection": {
      "type": "object",
      "patternProperties": {
        "^x-": {}
      },
      "additionalProperties": {
        "type": "object",
        "oneOf": [
          {
            "$ref": "#/definitions/RateLimitSection"
          }
        ]
      }
    },
    "AssemblyRateLimitsSection": {
      "type": "object",
      "patternProperties": {
        "^x-": {}
      },
      "additionalProperties": {
        "type": "object",
        "$ref": "#/definitions/AssemblyRateLimitSection"
      }
    },
    "AssemblyBurstLimitsSection": {
      "type": "object",
      "patternProperties": {
        "^x-": {}
      },
      "additionalProperties": {
        "type": "object",
        "$ref": "#/definitions/AssemblyBurstLimitSection"
      }
    },
    "AssemblyCountLimitsSection": {
      "type": "object",
      "patternProperties": {
        "^x-": {}
      },
      "additionalProperties": {
        "type": "object",
        "$ref": "#/definitions/AssemblyCountLimitSection"
      }
    },
    "BurstLimitsSection": {
      "type": "object",
      "additionalProperties": {
        "type": "object",
        "additionalProperties": false,
        "required": [
          "value"
        ],
        "patternProperties": {
          "^x-": {}
        },
        "properties": {
          "value": {
            "type": "string",
            "pattern": "^((\\d+)\\/(\\d+)?(second|minute))$"
          }
        }
      }
    },
    "GraphqlSection": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "per-query-policy": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "maximum-nesting": {
              "type": "integer"
            },
            "maximum-resolve-complexity": {
              "type": "integer"
            },
            "maximum-type-complexity": {
              "type": "integer"
            }
          }
        },
        "rate-limits": {
          "type": "object",
          "additionalProperties": {
            "type": "object",
            "oneOf": [
              {
                "$ref": "#/definitions/GraphqlRateLimitSection"
```

```
            }
          ]
        }
      }
    }

  },
  "GraphqlRateLimitSection": {
    "type": "object",
    "additionalProperties": false,
    "required": [
      "resolve-complexity",
      "type-complexity"
    ],
    "properties": {
      "resolve-complexity": {
        "$ref": "#/definitions/RateLimitSection"
      },
      "type-complexity": {
        "$ref": "#/definitions/RateLimitSection"
      }
    }
  },
  "BillingSection": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "billing": {
          "type": "string"
        },
        "currency": {
          "type": "string"
        },
        "price": {
          "type": "string",
          "pattern": "^\\d+(\\.\\d+)?$"
        },
        "period": {
          "type": "integer"
        },
        "period-unit": {
          "type": "string",
          "enum": [
            "month"
          ]
        },
        "trial-period": {
          "type": "integer"
        },
        "trial-period-unit": {
          "type": "string",
          "enum": [
            "day"
          ]
        }
      }
    }
  }
}
```

Completing the information section of your Product description

Provide users with information about your Product.

About this task

The information section of a YAML representation of a Product contains the Product's name and version. It can also include contact and license details.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.
An example information section for a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in An example YAML representation of a Product.

Procedure

To complete the information section of your Product description, complete the following steps.

1. Begin the section and provide a name, title, description, and version number; use the following syntax:

   ```
   info:
     version: Product_Version
     title: Product_Title
     name: Short_Name
     summary: Product_Description
   ```

   where:
   - *version* is the version number of the policy.
     Tip: The **version.release.modification** version numbering scheme is recommended, for example `1.0.0`.

- *Product_Title* is the title of the Product. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- *Short_Name* is the short name for the policy. It must be a single word and contain only alphanumeric characters, and the - (dash) and _ (underscore) characters. The name is case-sensitive, and should be 20 characters or fewer so that it can be displayed in the API Manager user interface of IBM API Connect.
- *Product_Description* is a short description of the policy. Any string can be used.

2. Optional: Under info, supply contact information; use the following syntax:

```
  contact:
    name: Contact_Name
    url: 'Contact_URL'
    email: Contact_email
```

where:
- *Contact_Name* is the name of the contact for this Product.
- *Contact_URL* is the URL for contact your organization.
- *Contact_email* is an email for contacting your organization.

3. Optional: Under info, add license information for your Product; use the following syntax:

```
  license:
    name: License_Name
    url: 'License_URL'
```

where:
- *License_Name* is the name of the license that applied to your Product.
- *License_URL* is the URL at which information about the license can be accessed.

4. Optional: Under info, add your terms of service; use the following syntax:

```
 termsOfService: Terms
```

where *Terms* is a string that details the terms of service for using your Product.

Results

You have completed the information section of your Product's YAML representation. It should have the following form:

```
info:
  version: Product_Version
  title: Product_Title
  name: Short_Name
  description: Product_Description
  contact:
    name: Contact_Name
    url: 'Contact_URL'
    email: Contact_email
  license:
    name: License_Name
    url: 'License_URL'
  termsOfService: Service_Terms
```

where all variables are as described in previously in this topic. The indentation must be as in the example.

Specifying a gateway type for your Product

Product definitions must specify which type of gateway the Product uses.

Before you begin

IBM® API Connect supports two gateway types: DataPower® Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see API Connect gateway types.

About this task

You can specify the gateway type for a Product.

Products can use only one type of gateway, and the APIs in the Product must use the same type of gateway. See Specifying a gateway type for an API definition.

Procedure

1. Determine the gateway type that is configured for the APIs in your Product. The Product gateway type must match the APIs gateway type.
   To review the APIs for your Product, see Referencing the APIs for your Product.

2. Edit the Product definition file (YAML source) to add the gateway type you want to use:
   - DataPower Gateway (v5 compatible):

```
gateways:
            datapower-gateway
```

- DataPower API Gateway

```
gateways:
            datapower-api-gateway
```

3. Ensure that the policies, including policy versions, in the APIs in your Product are supported by the gateway type. Review the list of policies on the execute page. If necessary, review the individual policy pages to determine the supported version number.

Related information

- DataPower API Gateway porting notes

Converting a Product YAML file to use DataPower API Gateway

IBM® API Connect now supports DataPower® API Gateway. You can optionally specify this gateway type in your Product YAML file.

Before you begin

IBM API Connect provides two gateway types, DataPower Gateway (v5 compatible) and DataPower API Gateway.

DataPower Gateway (v5 compatible) has been available with IBM API Connect for a number of years. The DataPower API Gateway is a new gateway that has been designed with APIs in mind, and with the same security focus as DataPower Gateway (v5 compatible).

For more information on how to choose which gateway type to use, see API Connect gateway types.

About this task

Product definitions can specify only one type of gateway. The type of gateway must match the gateway type that is used by the APIs that are included in the product. If you want to take advantage of the capabilities of DataPower API Gateway, you can easily modify your existing Product definitions to specify it as the gateway type.

Procedure

1. Verify that the APIs in the product specify DataPower API Gateway as the gateway type, and that the APIs have been converted for use with the DataPower API Gateway.
2. Open the Product YAML file for editing.
   For example:

```
product: '1.0.0'          .

info:
  name: testprod
  title: testprod
  version: 1.0.0
           .
           .
           .
```

3. Add a `gateways:` setting, and specify `datapower-api-gateway`.
   For example:

```
product: '1.0.0'          .

info:
  name: testprod
  title: testprod
  version: 1.0.0
gateways:
  - datapower-api-gateway
              .
              .
              .
```

4. Save the modified YAML file.

Specifying the visibility of your Product

How to set the availability of your Product by specifying its visibility and subscribability levels.

About this task

The visibility section of your Product's YAML representation determines who can view and subscribe to your Product in IBM® API Connect.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.
An example visibility section of a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in An example YAML representation of a Product.

Procedure

To complete the visibility section of your Product description, complete the following steps:

1. Title the section by adding **visibility:**
2. Under **visibility:**, specify who can **view** the Product; use the following syntax:

```
   view:
      enabled: View_Toggle
      type: View_Audience
      tags: View_Tag
      orgs: View_Organizations
```

   where
   - *View_Toggle* determines whether the Product is visible to anybody or not. It must be **true** or **false**. If **false**, the Product will not be visible in the Developer Portal.
   - *View_Audience* must be **public**, in which case the Product is visible to anybody who uses the Developer Portal, **authenticated**, in which case the Product is visible to anybody registered through the Developer Portal, or **custom**, in which case the Product is visible to a specified group of users. You can specify these custom users as one or more consumer organizations in the **orgs:** section, and as one or more consumer organization groups in the **tags:** section.
   - The *View_Tag* specifies the consumer organization groups that can view the Product if *View_Audience* is set to **custom**. If *View_Audience* is not set to **custom**, do not include **tags:**. If *View_Audience* is **custom**, you can specify groups in the following manner:

```
      tags:
         - Organization group 1
         - Organization group 2
```

     where the *Organization group* variables are the names of the groups that are to be allowed to view the Product. Note that when a Product is published, the **tags:** property name becomes **group_urls:**, and the content is changed to URL strings.
   - *View_Organizations* specifies the consumer organizations that can view the Product if *View_Audience* is set to **custom**. If *View_Audience* is not set to **custom**, do not include **orgs:**. If *View_Audience* is **custom**, you can specify organizations in the following manner:

```
      orgs:
         - Organization 1
         - Organization 2
```

     where the *Organization* variables are the names of the organizations that are to be allowed to view the Product. Note that when a Product is published, the **orgs:** property name becomes **org_urls:**, and the content is changed to URL strings.
3. Under **visibility:**, specify who can **subscribe** to your Product's Plans; use the following syntax:

```
   subscribe:
      enabled: Subscribe_Toggle
      type: Subscribe_Audience
      tags: Subscribe_Tag
      orgs: Subscribe_Organizations
```

   where:
   - *Subscribe_Toggle* must be **true** or **false**. If **false**, no developers will be able to subscribe to the Product.
   - *Subscribe_Audience* must be **authenticated**, in which case the Product's Plans can be subscribed to by anybody who is registered through the Developer Portal, or **custom**, in which case the Product's Plans can be subscribed to by a specified group of users.
   - The *Subscribe_Tag* specifies the consumer organization groups that can subscribe to the Product's Plans if *Subscribe_Audience* is set to **custom**. If *Subscribe_Audience* is not set to **custom**, omit it or set as **[]**. If *Subscribe_Audience* is **custom**, you can specify groups in the following manner:

```
      tags:
         - Organization group 1
         - Organization group 2
```

     where the *Organization group* variables are the names of the groups that are to be allowed to subscribe to the Product. Note that when a Product is published, the **tags:** property name becomes **group_urls:**, and the content is changed to URL strings.
   - *Subscribe_Organizations* specifies the organizations that can subscribe to the Product's Plans if *Subscribe_Audience* is set to **custom**. If *Subscribe_Audience* is not set to **custom**, omit it or set as **[]**. If *Subscribe_Audience* is **custom**, you can specify organizations in the following manner:

```
      orgs:
         - Organization 1
         - Organization 2
```

     where the *Organization* variables are the names of the organizations that are to be allowed to subscribe to the Product. Note that when a Product is published, the **orgs:** property name becomes **org_urls:**, and the content is changed to URL strings.
     Note: If a Product is to be available for subscription, it must already be visible. As a result, for the subscription audience to be **authenticated**, the view audience cannot be **custom**.

Results

You have completed the visibility section of your Product's YAML representation. It should have the following form:

```
visibility:
   view:
      enabled: View_Toggle
      type: View_Audience
      tags:
         - View_Tag_1
         - View_Tag_2
      orgs:
         - Organization_1
         - Organization_2

   subscribe:
      enabled: Subscribe_Toggle
```

```
    type: Subscribe_Audience
    tags:
       - Subscribe_Tag_1
       - Subscribe_Tag_2
    orgs:
       - Organization_1
       - Organization_2
```

where all variables are as described previously in this topic. The indentation must be as in the example.

What to do next

You can also define Product visibility at a Catalog or Space level. For more information, see Configuring Product visibility at a Catalog or Space level.

Related information

- Configuring Product visibility at a Catalog or Space level

Referencing the APIs for your Product

Detail the file paths for the APIs you want to include in your Product in IBM® API Connect.

About this task

Before an API can be included in a Plan, it must first be referenced in the APIs section of your Product description.

Procedure

Begin the APIs section and reference the APIs you want to include in your Product. The syntax that you use depends on which user interface you are using, as follows:

- API Designer:

  ```
  apis:
    API_1_NameAPI_1_Version
      $ref: API_1_File_Path
    API_2_NameAPI_2_Version
      $ref: API_2_File_Path
  ```

- API Manager:

  ```
  apis:
    API_1_NameAPI_1_Version
      name: 'API_1_Name:API_1_Version'
    API_2_NameAPI_2_Version
      name: 'API_2_Name:API_2_Version'
  ```

  where
  - the *API_n_Name* variables are the case-sensitive names of your APIs.
  - the *API_n_Version* variables are the versions of your APIs.
  - the *API_n_File_Path* variables are the file paths for the YAML files containing OpenAPI representations of your APIs.

The following example shows a complete sample APIs section:

```
apis:
  api11.0.0:
    name: 'api1:1.0.0'
  api21.0.0:
    name: 'api2:1.0.0'
  api31.0.0:
    name: 'api3:1.0.0'
```

For more information about creating OpenAPI definitions, see Creating an OpenAPI definition file.

The indentation must be as in the examples and the APIs must have been created before they can be referenced.

Results

You have referenced the APIs that are to be included in your Product. An example of a complete YAML representation of a Product can be found in An example YAML representation of a Product.

Describing Plans in your Product

Describe the Plans that you want to include in your IBM® API Connect Product and which APIs they will contain, as well as any rate limits that apply.

About this task

A Plan contains APIs and their operations. It can be used to implement rate limits and tailor visibility.

Note: If you are using more than one DataPower® server in a Gateway service, then to properly calculate API calls for rate limits the servers must be able to communicate with each other by using SLM peer groups, using either SLM unicast peering or SLM multicast peering depending on your network configuration. For more information, see SLM peering.
In addition, you can apply burst limits to your Plans, to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals. You can also set multiple rate limits per Plan and per operation, at second, minute, hour, day, and week time intervals.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.
Two example Plan descriptions from a YAML representation of a Product can be found at the end of this topic. An example of a complete YAML representation of a Product can be found in An example YAML representation of a Product.

Procedure

To describe your Plans, include APIs in them, and set rate limits for Plans or specific operations, complete the following instructions:

1. Begin the Plans section with `plans:`
2. Under Plans, begin the description of your first Plan by providing a name, description, and specifying whether approval is required for subscription requests; use the following syntax:

```
plans:
  Plan_Name:
    title: Plan_Title
    description: Plan_Description
    approval: Approval_Toggle
```

   where:
   - *Plan_Name* is the name of the Plan. It must be a single word and contain only alphanumeric characters, and the - (dash) and _ (underscore) characters. The name is case-sensitive, and should be 20 characters or fewer so that it can be displayed in the API Manager user interface.
   - *Plan_Title* is the title of the Plan.Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
   - *Plan_Description* is a description of your Plan.
   - *Approval_Toggle* must be either `true`, in which case approval is needed for subscription requests, or `false`, in which case subscriptions are automatically approved.
3. Optional: Under your Plan, add multiple burst and rate limits that will be shared across all the operations in the Plan; use the following syntax:
   Note: For information about rate limits and burst limits in API Connect, see Understanding rate limits for APIs and Plans.

```
    rate-limits:
      Name:
        value: Rate_Limit
        hard-limit: Limit_Toggle
      Name:
        value: Rate_Limit
        hard-limit: Limit_Toggle
    burst-limits:
      Name:
        value: Burst_Limit
```

   where:
   - *Name* is the name of the limit.
   - *Rate_Limit* is your rate limit. It can be for multiples of seconds, minutes, hours, days, or weeks, written as `second`, `minute`, `hour`, `day`, and `week` respectively; do not use the plural forms of the words. Use the syntax: **`1/2minute`**. If the time unit is singular, then it is not necessary to precede it with a number, for example, **`1/minute`**. If you do not want to apply a rate limit, set *Rate_Limit* as `unlimited`.
   - *Limit_Toggle* must be `true` or `false`. If *true*, API calls by a developer will fail if the rate limit is exceeded. This step is not necessary if you have set *Rate_Limit* to `unlimited`.
   - *Burst_Limit* is your burst limit. It can be for multiples of seconds or minutes, written as `second` or `minute`; do not use the plural forms of the words.

   Note:
   - Applying a rate limit at the Plan level creates a default rate limit that is shared across all the operations within the Plan. If you need to set specific rate limits for specific operations, you must set these within the operations themselves and these settings will override the setting at the Plan level.
   - The test application that is used by the API Manager test tool is not subject to rate limits if you have enabled automatic subscriptions for the Catalog in which you are testing. For more information, see Working with Catalogs
4. Under your Plan, specify which APIs are to be included.
   Reference APIs by name and version, as follows:

```
    apis:
      API_1_nameAPI_1_version: {}
      API_2_nameAPI_2_version: {}
```

   where
   - the *API_n_name* variables are the case-sensitive names of your APIs.
   - the *API_n_version* variables are the versions of your APIs.
5. Optional: If you want to include only a subset of an API's operations, list the operations that are to be included; use the following syntax:

```
    apis:
      API_nameAPI_version:
        operations:
          - operation: Operation_1_Verb
            path: Operation_1_Path
          - operation: Operation_2_Verb
            path: Operation_2_Path
```

   where
   - the *Operation_n_Path* variables are the paths of operations that is to be included. The dash is required for each new operation.
   - the *Operation_n_Verb* variables are the appropriate REST verbs for the operations.
6. Optional: If you want to specify multiple rate limits for a single operation; use the following syntax:

```
    apis:
      API_nameAPI_version:
        operations:
          - operation: Operation_Verb
```

```
            path: Operation_Path
            rate-limits:
              Name:
                value: Operation_Limit
                hard-limit: Operation_Limit_Toggle
              Name:
                value: Operation_Limit
                hard-limit: Operation_Limit_Toggle
```

where:

- *Operation_Limit* is the rate limit that you want to apply to your operation. It can be for multiples of seconds, minutes, hours, days, or weeks, written as `second`, `minute`, `hour`, `day`, and `week` respectively; do not use the plural forms of the words. Use the syntax: **`1/2minute`**. If the time unit is singular, then it is not necessary to precede it with a number, for example, **`1/minute`**. If you do not want to apply a rate limit, set *Operation_Limit* as `unlimited`.
- *Operation_Limit_Toggle* must be `true` or `false`. If *true*, API calls by a developer will fail if the rate limit is exceeded. This step is not necessary if you have set *Operation_Limit* to `unlimited`.

Results

You have described the Plans that are to be included in your Product. Your Plans section should be similar to the following scenarios.

If you have enforced a rate limit for your Plan and not specified individual operations, your Plans section should have the following form:

```
plans:
  Plan_Name:
    title: Plan_Title
    description: Plan_Description
    approval: Approval_Toggle
    rate-limits:
      Name:
        value: Limit
        hard-limit: Limit_Toggle
    apis:
      API_1_nameAPI_1_version: {}
      API_2_nameAPI_2_version: {}
```

If you have enforced rate limits at the Plan level and want to include two operations, with a separate rate limit one for one of the operations, your Plans section should have the following form:

```
plans:
  Plan_Name:
    title: Plan_Title
    description: Plan_Description
    approval: Approval_Toggle
    rate-limits:
      Name:
        value: API_Limit
        hard-limit: API_Limit_Toggle
    burst-limits:
      Name:
        value: Burst_limit
    apis:
      API_nameAPI_version:
        operations:
          - operation: Operation_1_Verb
            path: Operation_1_Path
            rate-limits:
              Name:
                value: Operation_Limit
                hard-limit: Operation_Limit_Toggle
          - operation: Operation_2_Verb
            path: Operation_2_Path
```

In both examples, variables are as in the previous steps and the indentation must be as presented.

The following example shows a complete sample Plans section:

```
plans:
  default:
    title: Default Plan
    description: Default Plan
    approval: false
    rate-limits:
      default:
        value: 100/hour
        hard-limit: false
  my-plan:
    title: my_plan
    rate-limits:
      Default rate-limit:
        value: 50/1hour
        hard-limit: false
    burst-limits:
      Default burst-limit:
        value: 10/1second
    apis:
      api21.0.0:
        operations:
          - operation: get
            path: /path2
      api31.0.0:
        operations:
          - operation: get
            path: /path3
```

```
            rate-limits:
              limit3:
                value: 10/1second
                hard-limit: true
              limit4:
                value: 100/1hour
                hard-limit: true
      api11.0.0: {}
```

An example YAML representation of a Product

Products in IBM® API Connect can be represented by using a YAML file in a similar fashion to how APIs can be represented by using OpenAPI.

The following code describes a complete Product, with sample values.

```
info:
  version: 1.0.0
  title: SampleProduct
  name: SampleProduct
  summary: This is a sample product
gateways:
  - datapower-gateway
product: 1.0.0
visibility:
  view:
    enabled: true
    type: public
    tags: []
    orgs: []
  subscribe:
    enabled: true
    type: authenticated
    tags: []
    orgs: []
plans:
  default:
    title: Default Plan
    description: Default Plan
    approval: false
    rate-limits:
      default:
        value: 100/hour
        hard-limit: false
  newplan:
    title: NewPlan
    rate-limits:
      low:
        value: 10/1hour
        hard-limit: true
    burst-limits:
      Default burst-limit:
        value: 10/1second
apis:
  SampleAPI:
    name: 'Sampleapi:1.0.0'
```

Using `x-ibm-languages` to create multilingual API and Product documentation

Create multilingual API and Product documentation by using the **`x-ibm-languages`** extension in your API and Product OpenAPI definitions.

You can scale your API initiatives to a global user base, while still maintaining a single API definition. Translations are custom configured and controlled, so that they can be tailored to both your API documentation and the API consumer. The extensions that are used in API Connect are added directly to the YAML file as **`x-ibm-languages`**. These extensions are then used in the Developer Portal, so that whatever language the Developer Portal is set to reflects the translations that are contained in the API and Product definitions.

Add the following syntax into your API or Product YAML file at every point that you require some translated text:

```
x-ibm-languages:
  item_name:
    language_code: translated_text
```

Where:

- ***`item_name`*** is the name of the item that you want to translate, for example **`summary`**.
- ***`language_code`*** is the ISO code for the translation language. Supported languages with their ISO codes are listed in the following table.

Table 1. List of supported language codes

Language	Code
Chinese, simplified	**`zh_cn`**
Chinese, traditional	**`zh_tw`**
Dutch	**`nl`**
English	**`en`**
French	**`fr`**

Language	Code
German	`de`
Italian	`it`
Japanese	`ja`
Korean	`ko`
Portuguese	`pt`
Spanish	`es`
Turkish	`tr`

- *translated_text* is the translated text that you want to be displayed for the item in the Developer Portal.

Note: The Developer Portal automatically assumes that the default language in the OpenAPI definition is English. If you want to use a different default language in the definition, you must provide both the English translation and the default language text in the **`x-ibm-languages`** extension sections of your OpenAPI definition, by using the following syntax:

```
x-ibm-languages:
  item_name:
    en: English_translated_text
    default_language_code: default_language_text
```

For an example of how to write an OpenAPI YAML file with French as the default language, see the Examples section.

Examples

An example API OpenAPI YAML file that contains the **`fr`** language extension, where English is the default language:

```
swagger: '2.0'
info:
  x-ibm-name: climbing-weather-api
  title: Climbing Weather API
  version: 1.0.0
  x-ibm-languages:
    title:
      fr: Climat d'escalade
schemes:
  - https
host: $(catalog.host)
consumes:
  - application/json
produces:
  - application/json
x-ibm-configuration:
  assembly:
    execute:
      - invoke:
          target-url: 'https://1234.com'
          title: 3 day forecast invocation
          cache-response: time-to-live
          cache-key: $(request.search)
  gateway: datapower-gateway
  enforced: true
  testable: true
  phase: realized
  cors:
    enabled: true
paths:
  /weather/forecast:
    get:
      summary: Retrieve the 3 day forecast for a location
      description: Retrieve the locations weather forecast descriptions for the next 3 days and nights
      operationId: getWeather
      x-ibm-languages:
        summary:
          fr: Récupérer la prévision de 3 jours pour un emplacement
        description:
          fr: Récupérer les descriptions des prévisions météo pour les 3 prochains jours et les nuits
      tags:
        - Weather
      parameters:
        - name: zip
          type: string
          in: query
          description: A 5 number zip code
          x-ibm-languages:
            description:
              fr: Un code postal à 5 numéros
        - name: country
          type: string
          in: query
          description: A 2 letter country code
          x-ibm-languages:
            description:
              fr: Un code de pays de 2 lettres
        - name: lat
          type: string
          in: query
          description: A latitude value between -90 and 90
          x-ibm-languages:
            description:
              fr: Une valeur de latitude entre -90 et 90
        - name: lon
```

```
          type: string
          in: query
          description: A longitude value between -180 and 180
          x-ibm-languages:
            description:
              fr: Une valeur de longitude entre -180 et 180
      responses:
        '200':
          description: Success
          x-ibm-languages:
            description:
              fr: Succès
        '400':
          description: Bad Request
          x-ibm-languages:
            description:
              fr: Mauvaise Demande
        '408':
          description: Request Timeout
          x-ibm-languages:
            description:
              fr: Délai de délai de demande
        '500':
          description: Internal Server Error
          x-ibm-languages:
            description:
              fr: Erreur Interne du Serveur
basePath: /
tags:
  - name: Weather
    description: Sample API to get weather forecast data
    x-ibm-languages:
      description:
        fr: Exemple d'API pour obtenir des données météorologiques
securityDefinitions:
  client-secret:
    type: apiKey
    description: ''
    in: header
    name: X-IBM-Client-Secret
  client-id:
    type: apiKey
    description: ''
    in: header
    name: X-IBM-Client-Id
security:
  - client-secret: []
    client-id: []
```

An example Product OpenAPI YAML file that contains the `fr` language extension, where English is the default language:

```
product: 1.0.0
info:
  name: climbing-weather
  title: Climbing Weather
  version: 1.0.0
  description: This is a product about weather.
  x-ibm-languages:
    title:
      fr: Météo traduite
    description:
      fr: C'est un produit sur la météo.
    termsOfService:
      fr: Quid ergo aliud intellegetur en francais.
  contact:
    name: Ralph Renli
    email: ralph.renli@example.com
    url: 'https://weather.example.com/climbing/info'
  license:
    name: MIT License
    url: 'https://choosealicense.com/licenses/mit/'
    x-ibm-languages:
      name:
        fr: License de MIT
  termsOfService: Quid ergo aliud intellegetur nisi uti ne quae pars naturae neglegatur? Quis est tam dissimile homini. De
quibus cupio scire quid sentias.
  categories:
    - Portal/Testing/Language
    - Portal/Testing/Language/UTF8
    - Portal/Testing/Weather
visibility:
  view:
    enabled: true
    type: public
    tags: []
    orgs: []
  subscribe:
    enabled: true
    type: authenticated
    tags: []
    orgs: []
apis:
  climbing-weather:
    id: 593f972be4b06fb4e0dce879
plans:
  a-call-a-day:
```

```
    title: A call a day
    description: "One call every day. That's it!"
    x-ibm-languages:
      title:
        fr: "Un appel par jour"
      description:
        fr: "Un appel tous les jours. C'est tout!"
    apis: {}
    rate-limits:
      One Call Per Day:
        hard-limit: true
        value: 1/1day
  ten-calls-per-day:
    title: 10 Calls a day
    description: '10 times more calls than the next best competitor plan!'
    x-ibm-languages:
      title:
        fr: "10 appels par jour"
      description:
        fr: "10 fois plus d'appels que le prochain meilleur plan concurrent!"
    apis: {}
    rate-limits:
      10 calls per day:
        hard-limit: true
        value: 10/1minute
  11-calls-every-day:
    title: 11 Calls a day
    description: 'Need just one more call? Then this is the plan for you!'
    x-ibm-languages:
      title:
        fr: "11 appels par jour"
      description:
        fr: "Vous n'avez besoin que d'un appel supplémentaire? Alors c'est le plan pour vous!"
    apis: {}
    rate-limits:
      11 calls per day:
        hard-limit: true
        value: 11/1day
  25-calls-per-day:
    title: 25 Calls per day
    description: So you want even more calls?
    apis: {}
    x-ibm-languages:
      title:
        fr: "25 appels par jour"
      description:
        fr: "Vous voulez donc encore plus d'appels?"
    rate-limits:
      25 calls per day:
        hard-limit: true
        value: 25/1day
  100-calls-per-day:
    title: 100 Calls every day
    description: 'You get 100 calls each and every day!'
    x-ibm-languages:
      title:
        fr: "100 appels par jour"
      description:
        fr: "Vous recevez 100 appels chaque jour"
    apis: {}
    rate-limits:
      100 Calls per day:
        hard-limit: true
        value: 100/1day
```

An example of how to write an OpenAPI YAML file with French as the default language:

```
info:
  name: climat-d-escalade
  title: Climat d'escalade
  version: 1.0.0
  x-ibm-languages:
    title:
      en: Climbing Weather API
      fr: Climat d'escalade
...
```

This example shows that both the default French language, and the English language translation, must be provided in the **`x-ibm-languages`** section.

Related information

- Expand your API Initiatives Globally with Multi-Lingual Support of API and Product Definitions

Using `x-example` to control the examples displayed in the Developer Portal

OpenAPI does not support the use of **`example`** attributes on parameters, only on request and response objects and their properties. To allow API Developers to be able to control the examples displayed in the portal you can use **`x-example`** in OpenAPI parameters.

In this example, you cannot use **`example`** as it isn't supported by the OpenAPI spec, but you can use **`x-example`**.

```
- name: city
  in: query
  required: false
  type: string
  maxLength: 50
  description: "Location"
  x-example: "New York"
```

An **example** attribute is honored where set and allowed by the spec. If **example** is not found, **default** is honored where set and allowed by the spec. An **x-example** attribute is honored if found, if not found, a random example to match the schema is generated.

If the spec allows you to use **example**, then you should use it in preference to **x-example**.

Using x-embedded-doc to add additional documentation to products and APIs

You can use **x-embedded-doc** to add additional documentation to a product or an API as part of their definition.

You can embed additional documentation pages within the api swagger documentation. These pages can be base64 encoded strings or markdown strings of the html content of the page. These pages show up in the navigation, on the left side, underneath the overview for that product or API, and above the paths. Any styling that is needed to theme the documentation can be included in the Developer Portal custom theme.

Embedded docs for products and APIs

You can embed documentation for individual products and APIs by specifying an additional **x-embedded-doc** object in the product or API specification.

Field name	Type	Description
name	string	Required, the name of the embedded document
title	string	Required, The display title of the embedded document
format	string	Optional, the format of the embedded document content value. Values are **b64html** or **md**. Default value is **md**.
content	string	The base64 encoded html of page, or, the markdown string.
docs	x-embedded-doc	Additional embedded documentation. If no content is specified for parent, then the first child is the content that is shown.

Hierarchies of documents are supported by nesting of further doc properties. Language support is available by using **x-ibm-languages**.

Examples

Documentation can be added to an api, by using the following JSON:

```
"x-embedded-doc": [
   {
     "name": "introduction",
     "title": "Introduction",
     "format": "md",
     "content": "# some markdown"
   },
   {
     "name": "concepts",
     "title": "Concepts",
     "format": "md",
     "content": "## more markdown"
   },
   {
       "name": "authentication",
       "title": "Authentication",
       "docs": [{
             "name": "auth_clientid_secret",
             "title": "Obtaining a Client ID and Secret",
             "format": "md",
             "content": "### even more markdown"
             },
             {
               "name": "auth_bearertoken",
               "title": "Getting and Using a Bearer Token",
               "format": "md",
               "content": "#### yet more markdown"
             }
             ]
   }
 ]
```

Alternatively, by using base 64 encoded HTML:

```
"x-embedded-doc": [
  {
    "name": "introduction",
    "title": "Introduction",
    "format": "b64html",
    "content": "PGFydGljbGUgaWQ9ImludHJvZHVjdGlvbiIgY2xhc3M...."
  },
  {
    "name": "concepts",
    "title": "Concepts",
    "format": "b64html",
    "content": "PGFydGljbGUgaWQ9ImNvbmNlcHRzIiBjbGFzcz0icGFnZSI....."
  },
  {
```

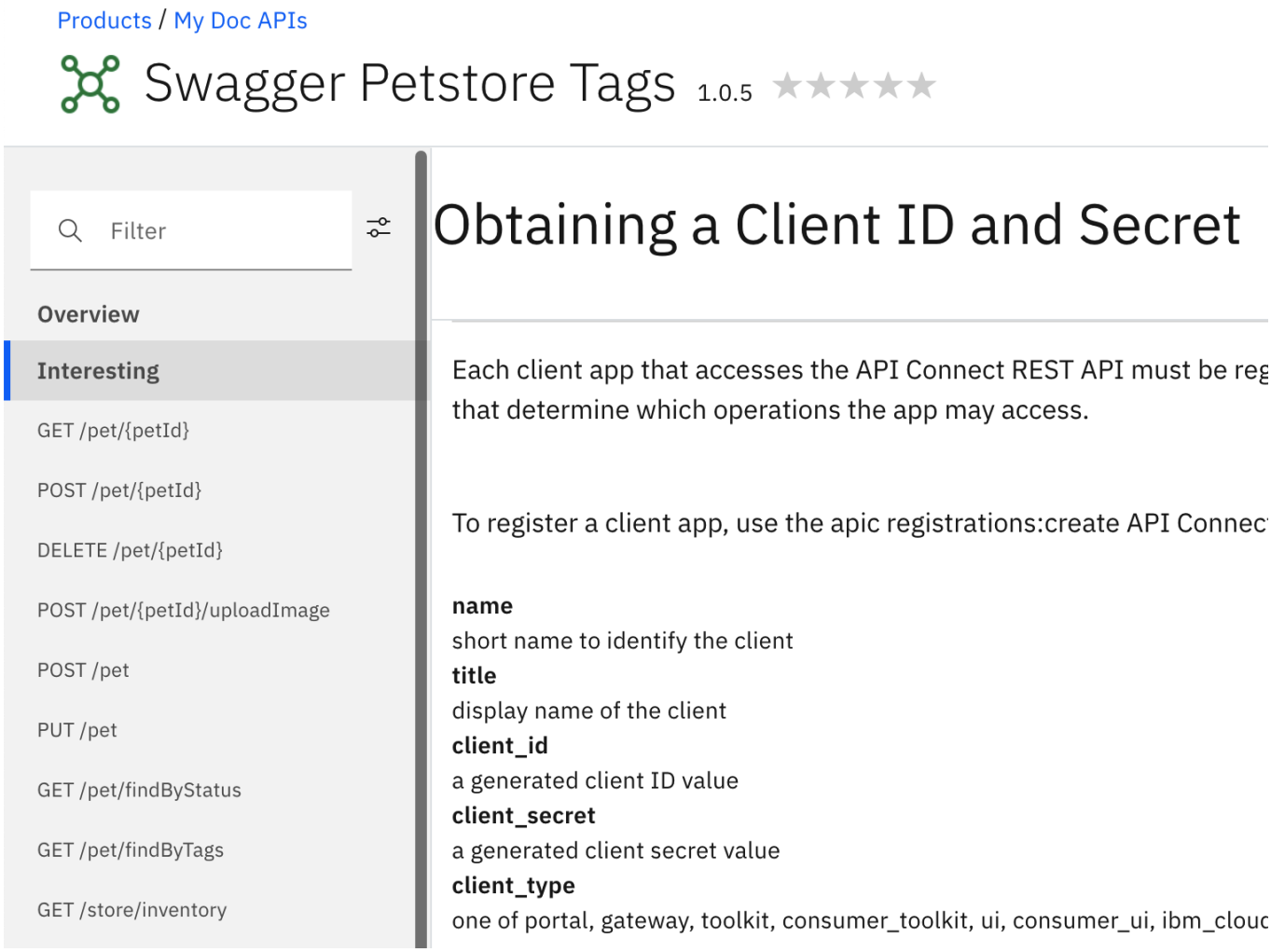
```
    "name": "authentication",
    "title": "Authentication",
    "docs": [{
            "name": "auth_clientid_secret",
            "title": "Obtaining a Client ID and Secret",
            "format": "b64html",
            "content": "PGFydGljbGUgaWQ9ImNsaWVudC1pZC1zZWNyZX....."
            },
            {
            "name": "auth_bearertoken",
            "title": "Getting and Using a Bearer Token",
            "format": "b64html",
            "content": "ICA8YXJ0aWNsZSBpZD0iYmVhcmVyLXRva2VuIiBjbGFz....."
            }
          ]
      }
]
```

Or mixed formats:

```
"x-embedded-doc": [
  {
    "name": "introduction",
    "title": "Introduction",
    "format": "b64html",
    "content": "PGFydGljbGUgaWQ9ImludHJvZHVjdGlvbiIgY2xhc3M...."
  },
  {
    "name": "concepts",
    "title": "Concepts",
    "format": "md",
    "content": "# some markdown"
  }
]
```

An example of some `x-embedded-doc` code and how it might render:

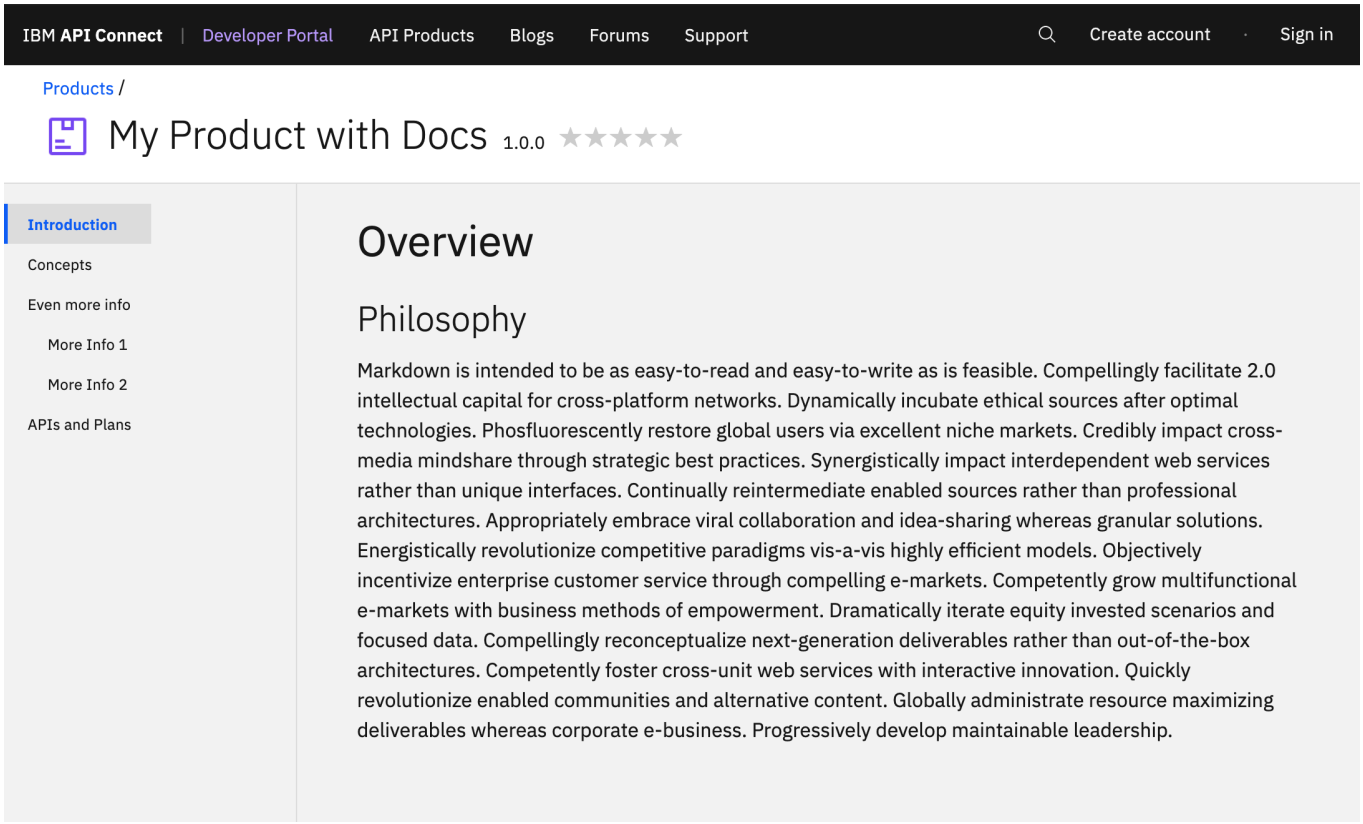
```
x-embedded-doc:
  - name: document
    title: Interesting
    format: b64html
    content: >-
      PGFydGljbGUgaWQ9ImNsaWVudC1pZ...=
securityDefinitions:
  api_key:
    type: apiKey
    in: header
    name: api_key
```



An example of some **x-embedded-doc** code into a product and how it might render:

```
x-embedded-doc:
  - name: introduction
    title: Introduction
    format: md
    content: >-
      ## Overview

      ### Philosophy

      Markdown is intended to be as easy-to-read and easy-to-write as is
      feasible. Compellingly facilitate 2.0 intellectual capital for
      cross-platform networks.  Dynamically incubate ethical sources after
      optimal technologies. Phosfluorescently restore global users via excellent
      niche markets. Credibly impact cross-media mindshare through strategic
      best practices. Synergistically impact interdependent web services rather
      than unique interfaces.

      Continually reintermediate enabled sources rather than professional
      architectures. Appropriately embrace viral collaboration and idea-sharing
      whereas granular solutions. Energistically revolutionize competitive
      paradigms vis-a-vis highly efficient models. Objectively incentivize
      enterprise customer service through compelling e-markets. Competently grow
      multifunctional e-markets with business methods of empowerment.

      Dramatically iterate equity invested scenarios and focused data.
      Compellingly reconceptualize next-generation deliverables rather than
      out-of-the-box architectures. Competently foster cross-unit web services
      with interactive innovation. Quickly revolutionize enabled communities and
      alternative content. Globally administrate resource maximizing
      deliverables whereas corporate e-business.

      Progressively develop maintainable leadership.
  - name: concepts
    title: Concepts
    format: md
    content: '## some more markdown'
  - name: moreinfo
    title: Even more info
    docs:
      - name: moreinfo1
        title: More Info 1
        format: md
```

```
      content: '### even more markdown'
    - name: moreinfo2
      title: More Info 2
      format: md
      content: '#### yet more markdown'
```



Related reference

- Using x-ibm-languages to create multilingual API and Product documentation

Using `x-pathalias` to give consistent URLs for products and APIs

You can use **`x-pathalias`** to assign a URL to a product or an API.

The information section of a YAML representation of a Product or API can include **`x-pathalias`**. This option can be useful as it allows the same URL to be used on different revisions of your product. It also allows the use of a vanity URL, which might better represent the branding of your site, or make the URL easier to pronounce.

In this example, you set an API called `Climbing weather API` in product `Weather watch` to have a URL of ***`portalURL`*`/product/weather/api/climbing`**.

Product information:

```
info:
  version: 1.0.0
  title: Weather watch
  name: weather-watch
  x-pathalias: weather
```

API information:

```
info:
  title: Climbing Weather API
  x-ibm-name: climbing-weather-api
  version: 1.0.0
  x-pathalias: climbing
```

Related tasks

- Editing an OpenAPI 2.0 API definition

Related information

- Creating and configuring Catalogs

Validating the YAML or JSON definition of an API or Product

You can validate a YAML or JSON definition by using the IBM® API Connect developer toolkit.

Before you begin

To complete the steps that are described in this topic, you must have installed the developer toolkit. For more information, see Installing the toolkit.

Procedure

To perform validation by using the developer toolkit, enter the following command:

```
apic validate filename
```

where *filename* is the file name of the API definition file you want to validate.

- Include `--product-only` to validate only a Product definition and not any APIs that it references.
- Include `--no-extensions` to validate only the default OpenAPI section of the API and none of its extensions.

Note:

- If the OpenAPI file that defines your API uses a **`$ref`** field to reference a fragment of OpenAPI code that is defined in a separate file, the **`$ref`** field is replaced with the contents of the target file before the draft API is created with the **apic drafts:validate** command. For more information, see Using $ref to reuse code fragments in your OpenAPI files.
- Products that contain an API with a Swagger property using **regex** that include lookahead assertions, such as "**`(?`**" cannot be validated or published. An error message is returned. For example:

  ```
  Product has not been published!
  The multipart 'openapi' field contains an OpenAPI definition with validation errors.
      definitions.properties.pattern Does not match format 'regex' (context: (root).definitions.properties.pattern, line: 0,
  col: 0)
  400
  ```

Related concepts

- Creating an OpenAPI definition file

Creating and using API and Product definitions templates

You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form **`{{variable-name}}`** that are substituted with values when you create the API or Product definition.

Before you begin

To complete the steps that are described in this topic, you must have installed the IBM® API Connect developer toolkit. For more information, see Installing the toolkit.

Procedure

1. Create a Product or API definition template, either from scratch, or by copying one of the examples provided in API and Product definition template examples. The template file must have **`.hbs`** filename extension, and may contain any of the handlebars variables described in Template variables for API and Product definitions.
2. To create a Product or API definition from a template, enter the following command:

   ```
   apic create:[api | product] --template template_file --title [api_title | product_title] options
   ```

   where *template_file* is the template **`.hbs`** file to use, *api_title* or *product_title* is the title of the API or Product to create, and *options* is any additional command-line options. The path to the template file can be either an absolute path or relative to the location where the command is executed.

   When the Product or API definition is created, the value of each command-line option is substituted for the corresponding handlebars template variable. For example, the value of the required **`--title`** option is substituted for the **`info.title`** field in the template file. The command creates an API or Product definition YAML file with the name specified in the **`--name`** option. If you don't supply the **`--name`** option, the command derives the name of the API or Product YAML file from the specified title by down-casing the title and replacing spaces with dashes.

Related reference

- Template variables for API and Product definitions
- API and Product definition template examples

Working with global policies

Use global policies to configure policy assemblies that are called just before, or just after, each of your API assemblies is called, or when an error is thrown. You can upload global policies into each of the gateway services in your Catalogs, and then designate, for each gateway service, which global policy should be called before an API

assembly is called, after an API assembly is called, or if an error occurs when an API assembly is called. The designated global policies are applied to all the APIs that are deployed to the associated gateway service.

Prerequisites

IBM® DataPower® Gateway version that is compatible with your deployed version of API Connect.

To complete the global policy management tasks described in this topic, you must be the owner of the provider organization, or be assigned a role that has the **`Settings:Manage`** permission. The pre-supplied Administrator role has this permission by default; if you are assigned a custom role it must have this permission. For more information, see Creating custom roles.

Overview

For each of the gateway services in a Catalog, a global policy that you want to be called before any API assembly is called is designated as a *pre-request* global policy, and a global policy that you want to be called after any API assembly is called is designated as a *post-response* global policy. A policy that you want to be called when an error is thrown is called an *error* global policy. You can upload as many global policies into a gateway service as you want, but you can designate at most one global policy to be pre-request, one to be post-response, and one to be an error, at any one time.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
Note:

- If you are using the DataPower Gateway (v5 compatible), there can be only one **`proxy`** policy across the combined pre-request, API, and post-response assembly flows.
- Catalog properties, as described in Creating and configuring Catalogs, and API properties, as described in Setting API properties, are not supported with global policies. Therefore, if you use such properties in a global policy, they are **not** replaced with the values specified in the API or Catalog property definitions.

A global policy is defined in a .yaml file. The global policy file specifies the policy assembly that you want to call. A global policy file has the following structure:

```
global-policy: 1.0.0

info:
    name: global_policy_name
    title: global_policy_title
    version: global_policy_version

gateways:
    - gateway_type

assembly:
    - policy_assembly_to_be_called
```

where *gateway_type* has the value **`datapower-gateway`** if you want to use the global policy in a DataPower Gateway (v5 compatible), and **`datapower-api-gateway`** if you want to use the global policy in a DataPower API Gateway. For more information on the different types of gateway, see API Connect gateway types.
Note: Error global policies are supported only by the DataPower API Gateway, not by the DataPower Gateway (v5 compatible).
The **`assembly`** section for a pre-request or post-response global policy can contain any valid API assembly flow. For more information, see the execute section.

The **`assembly`** section for an error global policy can only contain a **`catch`** extension in the assembly, an **`execute`** section is not valid. For details of the **`catch`** extension, see catch.

Sample global policy .yaml file for pre-request or post-reponse

```
global-policy: 1.0.0

info:
  name: sample-policy
  title: Sample Policy
  version: 2.0.0

gateways:
  - datapower-api-gateway

assembly:
  execute:
    - invoke:
        target-url: http://myhost.com/mytarget
        version: 2.0.0

  catch:
    - errors:
        - ConnectionError
      execute:
        - activity-log:
            title: activity-log
            content: activity
            error-content: payload
            version: 2.0.0
```

Sample error global policy .yaml file

Note: An error global policy .yaml file can contain only a **`catch`** extension in the assembly.

```
global-policy: 1.0.0
```

```
info:
  name: sample-error-policy
  title: Sample Error Policy
  version: 2.0.0

gateways:
  - datapower-api-gateway

assembly:
  catch:
    - errors:
        - ConnectionError
      execute:
        - activity-log:
            title: activity-log
            content: activity
            error-content: payload
            version: 2.0.0
    - errors:
        - SOAPError
      execute:
        - xslt:
            version: 2.0.0
            title: xslt
            input: true
            source: |-
              <!-- Placeholder -->
              <xsl:stylesheet xmlns:xsl=''/>
```

Logging in to the Management server

You work with global policies by using developer toolkit CLI commands. Before you can run the commands, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

Uploading a global policy to a gateway service

You upload a global policy to a gateway service by using the **apic global-policies:create** command. The command has the following format:

```
apic global-policies:create --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name --server mgmt_endpoint_url --scope scope [--space space] filename
```

where:

- *catalog_name* is the value of the `name` property of the Catalog that contains the gateway service to which you want to upload the global policy.
- *gateway_service_name* is the value of the `name` property of the gateway service.
- *organization_name* is the value of the `name` property of the provider organization that contains the Catalog.
- *mgmt_endpoint_url* is the platform API endpoint URL.
- *scope* has one of the following values:
  - `catalog` if the Catalog does not have Spaces enabled.
  - `space` if the Catalog has Spaces enabled. If you specify `space` for the `--scope` parameter you must also supply the `--space` parameter to specify the Space that contains the gateway service.
- (optional) *space* is the name of the Space that contains the gateway service. The `--space` parameter is required if the Catalog has Spaces enabled, in which case you must also include `--scope space` in the command.
- *filename* is the name of the global policy .yaml file that you want to upload.

Note: If Spaces are enabled in a Catalog, a global policy that you apply to one Space is applied to **all** Spaces; you cannot apply a global policy to an individual Space in the Catalog. Any subsequent updates are also applied to all Spaces.
You can confirm the global policies that have been uploaded by using the **apic global-policies:list-all** command, which has the following format:

```
apic global-policies:list-all --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name --server mgmt_endpoint_url --scope scope [--space space]
```

For reference details of the `apic global-policies` commands, see apic global-policies.

Designating the pre-request global policy

To designate the pre-request policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

   ```
   apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
   --server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
   ```

   where *policy_name*:*policy_version* specifies the name and version number of the required global policy, as defined in the global policy `.yaml` file that was originally uploaded to the gateway service.
   The value of the URL is written to a file called GlobalPolicy.yaml.

2. Edit the GlobalPolicy.yaml file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

   ```
   global_policy_url: >-
     https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
   policies/policy_id
   ```

3. Designate the global policy to be the pre-request policy by using the following command:

   ```
   apic global-policy-prehooks:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
   organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
   ```

To replace the designated pre-request global policy with another policy, repeat steps 1 and 2, then use the `apic global-policy-prehooks:update` command. To remove the pre-request global policy designation, use the `apic global-policy-prehooks:delete` command.
For reference details of the `apic global-policy-prehooks` commands, see apic global-policy-prehooks.

By default, whenever an API is called, a series of policies, called the *preflow* policies, are invoked prior to the policy assembly configured in the API definition. However, by using a pre-request global policy you can, if required, fully customize the behavior of these policies, controlling if and when they are invoked, and with what customization. For more information, see Customizing the preflow policies.

Designating the post-response global policy

To designate the post-response policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

   ```
   apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
   --server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
   ```

   where *policy_name*:*policy_version* specifies the name and version number of the required global policy, as defined in the global policy `.yaml` file that was originally uploaded to the gateway service.
   The value of the URL is written to a file called GlobalPolicy.yaml.

2. Edit the GlobalPolicy.yaml file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

   ```
   global_policy_url: >-
     https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
   policies/policy_id
   ```

3. Designate the global policy to be the post-response policy by using the following command:

   ```
   apic global-policy-posthooks:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
   organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
   ```

To replace the designated post-response global policy with another policy, repeat steps 1 and 2, then use the `apic global-policy-posthooks:update` command. To remove the post-response global policy designation, use the `apic global-policy-posthooks:delete` command.
For reference details of the `apic global-policy-posthooks` commands, see apic global-policy-posthooks.

Designating the error global policy

The error global policy is applied to an error caught in the API pre-request, assembly, and post-response stages unless any of those stages has its own error rule defined for the same error, in which case that error rule takes priority.

To designate the error global policy for a gateway service, complete the following steps:

1. Write the URL of the required global policy to a .yaml file by using the following command:

   ```
   apic global-policies:get --catalog catalog_name --configured-gateway-service gateway_service_name --org organization_name
   --server mgmt_endpoint_url --scope scope [--space space] policy_name:policy_version --fields url
   ```

   where *policy_name*:*policy_version* specifies the name and version number of the required global policy, as defined in the global policy `.yaml` file that was originally uploaded to the gateway service.
   The value of the URL is written to a file called GlobalPolicy.yaml.

2. Edit the GlobalPolicy.yaml file and replace the string `url` with `global_policy_url`. The resulting file has the following format:

   ```
   global_policy_url: >-
     https://server_host_name/api/catalogs/catalog_id/configured-gateway-services/gateway_service_id/global-
   policies/policy_id
   ```

3. Designate the global policy to be the error policy by using the following command:

   ```
   apic global-policy-errors:create --catalog catalog_name --configured-gateway-service gateway_service_name --org
   organization_name --server mgmt_endpoint_url --scope scope [--space space] GlobalPolicy.yaml
   ```

To replace the designated error global policy with another policy, repeat steps 1 and 2, then use the `apic global-policy-errors:update` command. To remove the error global policy designation, use the `apic global-policy-errors:delete` command.

For reference details of the **`apic global-policy-errors`** commands, see apic global-policy-errors.

Customizing the preflow policies

By default, whenever an API is called, a series of policies, called the *preflow* policies, are invoked prior to the policy assembly configured in the API definition. However, by using a global policy you can, if required, fully customize the behavior of these policies, controlling if and when they are invoked, and with what customization.

Before you begin

- For details on how to configure and deploy a global policy, see Working with global policies.
- For details on how to avoid authentication errors on a WSDL action, see the IBM troubleshooting tech note, API Connect HTTP GET ?WSDL retrieval returns "401 - Unauthorized Invalid client id or secret".

Default preflow policies

The following preflow policies are invoked by default:

- **`cors`**: handles cross-origin resource sharing (CORS) requests for the API; invoked if CORS is enabled in the API definition settings.
- **`wsdl`**: handles WSDL requests.
- **`html-page`**: for a GraphQL API, if the GraphQL editor option is enabled, returns the GraphQL HTML page.
- **`client-identification`**: examines the API key credentials that are carried in the API request and matches the API Plan through which the target API is made available to the client.
- **`ratelimit`**: enforces the rate limit scheme that is configured for the matching API Plan. When the rate limit is reached, the request is rejected.
- **`security`**: performs the authentication and authorization checks that are required by the target API and operation. if the security requirement is not fulfilled, the request is rejected.

Configuring a global policy to customize the preflow policies

To customize the preflow policies, complete the following steps:

1. add the following property to the **`info:`** section of your global policy .yaml file:

   `mode: *mode_type*`

   Where ***`mode_type`*** is one of the following options:
   - **`after-builtin`** - this means that your preflow policies are applied after the default preflow policies.
   - **`before-builtin`** - this means that your preflow policies are applied before the default preflow policies.
   - **`full-custom`** - this means that you must explicitly add the preflow policies to the assembly of your global policy. You can, however, remove, reposition, or replace any of the preflow policies as required.
2. Configure the policies in your global policy assembly as required.
   An **`after-builtin`** global policy example:

```
global-policy: 1.0.0
info:
  name: foo-example
  title: foo-example
  version: 2.0.0
  mode: after-builtin
gateways:
  - datapower-api-gateway
assembly:
  execute:
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: message.headers.X-Flexible-Preflow
            value: 'Hello reboot global policy 1 on catalog 1 this policy will be executed after all default builtins'
            type: string
```

   A **`before-builtin`** global policy example:

```
global-policy: 1.0.0
info:
  name: foo-example
  title: foo-example
  version: 2.0.0
  mode: before-builtin
gateways:
  - datapower-api-gateway
assembly:
  execute:
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: message.headers.X-Flexible-Preflow
            value: 'Hello reboot global policy 1 on catalog 1 this policy will be executed before all default builtin
policies'
            type: string
```

   A **`full-custom`** global policy example:

```
global-policy: 1.0.0
info:
  name: preflow-4-actions
  title: preflow-4-actions
  version: 2.0.0
  mode: full-custom
gateways:
  - datapower-api-gateway
assembly:
  execute:
    - cors:
        version: 2.0.0
        title: cors-in-preflow
    - wsdl:
        title: default-wsdl
    - html-page:
        title: html-page-in-preflow
        output: message
        version: 2.0.0
    - helloworld:
        version: 1.0.0
        title: helloworld
    - client-identification:
        version: 2.0.0
        title: ci-in-preflow
    - ratelimit:
        version: 2.0.0
        title: assembly-rt-in-preflow
        source: plan-default
    - security:
        version: 2.0.0
        title: sec-in-preflow
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: message.headers.X-Flexible-Preflow
            value: 'Hello reboot with 4 actions'
            type: string
```

3. Deploy your global policy as a pre-request global policy.

Reference

Reference information for developing your APIs in API Connect, including context variables, and template variables.

Note: For product-wide reference material, including command-line tool and REST APIs documentation, see Reference information for IBM® API Connect.

- **API Connect context variables**
  List of IBM API Connect context variables that you can reference when defining default parameter values in an assembly operation, or by using the `getContext()` function when defining a policy.
- **OAuth context variables**
  You can customize the OAuth security flow in a native OAuth provider by adding multiple OAuth policies to the assembly. Each OAuth policy takes input from context variables and writes output to context variables. By using these OAuth context variables you can manipulate the original request, and process the output from an OAuth policy.
- **Error cases supported by assembly catches**
  Several error cases that can be returned by the assembly are available to the catch function.
- **API and Product definition template examples**
  You can use template files when creating API and Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition. This topic provides the default Handlebar templates used by `apic` to create Products and APIs as examples you can use to copy and customize for your own use.
- **Template variables for API and Product definitions**
  You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form `{{variable-name}}` that are substituted with values when you create the API or Product definition.

Related information

- IBM API Connect overview

API Connect context variables

List of IBM® API Connect context variables that you can reference when defining default parameter values in an assembly operation, or by using the `getContext()` function when defining a policy.

The following tables are presented:

- General context variables.
- OAuth context variables - DataPower Gateway (v5 compatible).
- Application certificate context variables.
- API activity logging context variables.
- GraphQL context variables.

For more information about implementing an assembly component, see Including elements in your OpenAPI 2.0 API assembly, Including elements in your OpenAPI 2.0 API assembly, and for information about how to reference context variables in API Connect see Variable references in API Connect and Using context variables in GatewayScript and XSLT policies with the DataPower API Gateway.

For more information about creating a user-defined policy, see Authoring policies.

General context variables

Note:

- For plan variables (such as `plan.name` or `plan.version`), plan information is available only when the requested operation requires identification and the client passes the authentication check.
- If you deploy your API to the DataPower® Gateway (v5 compatible) then, with the exception of client ID and client secret, the passing of form input as a parameter into an API is not supported. This restriction does not apply if you deploy your API to the DataPower API Gateway.

Table 1. API Connect context variables

Name	Description	Permission
API Gateway only `api.catalog.id`	The ID of the Catalog in which the API is published.	Read/write
API Gateway only `api.catalog.name`	The name of the Catalog in which the API is published.	Read/write
API Gateway only `api.catalog.path`	The path segment that represents this Catalog.	Read/write
`api.endpoint.address`	The address of the API Gateway endpoint.	Read/write
`api.endpoint.hostname`	The host name of the API Gateway endpoint, as requested by the application.	Read/write
API Gateway only `api.id`	The identifier of the API.	Read/write
`api.name`	The name of the API; this corresponds to the `x-ibm-name` field in the OpenAPI definition for the API.	Read/write
`api.operation.id`	The ID of the operation.	Read/write
`api.operation.path`	The path of the operation.	Read/write
`api.org.id`	The organization ID of the API provider.	Read/write
`api.org.name`	The organization short name of the API provider.	Read/write
`api.properties.`*`propertyname`*	The name of a custom API property. Property values are Catalog specific. Note: • You have write permission to a custom property **only** from the user interface, **not** from GatewayScript.	Read/write
`api.root`	The API base path.	Read/write
`api.type`	The API type; REST or SOAP.	Read/write
`api.version`	The version string of the API.	Read/write
`client.app.id`	The client ID or application key that is received on the request.	Read/write
API Gateway only `client.app.'lifecycle-state'`	The lifecycle status of the calling client application. The possible values are as follows: • `DEVELOPMENT` • `PRODUCTION` (default)	Read/write

API Gateway only `client.app.metadata.`*`key`*	The string value of an application metadata key, where *key* is the name of the key.
	You can add metadata keys to an application by using either the **apic apps:create** or the **apic apps:update** command; you include the metadata keys in the configuration file parameter that is passed to the command.
	For example:
	`apic apps:create myapp.yaml --server myserver.com --org myorg --catalog mycatalog --consumer-org mycorg`
	where myapp.yaml contains the following:
	`name: myapp`
	`title: My test application`
	`metadata:`
	`  key1: value1`
	`  key2: value2`
	`  key3: value3`
	`  key4: value4`
	`  key5: value5`
	You can then retrieve the value of a metadata key in an API assembly policy by using a context variable such as the following:
	`client.app.metadata.key3`
	Note that adding application metadata might impact gateway trans performance.
		Read/write

`client.app.name`	The name of the application that is identified as having issued the request.	Read/write
`client.app.secret`	The client secret that is received in the request.	Read/write
`client.app.type`	The type of the client application that issued the request.	Read/write
`client.org.id`	The unique identification key of the organization that owns this application.	Read/write
`client.org.name`	The name of the organization that owns this application.	Read/write
API Gateway only `client.result`	The result of the client security policy, which is `SUCCESS` or `FAILURE`.	Read/write

Name	Description	Permission
API Gateway only `client.third_party.type`	The type of user registry used for third-party authentication of the extracted client credentials. The possible values are `LDAP` and `auth-url`.	Read/write
API Gateway only `client.third_party.headers`	The array of headers added to the request that was sent to that API authentication URL during third-party authentication.	Read/write
API Gateway only `client.third_party.response.authenticated`	The third-party authentication results. The possible values are as follows: • `true`: the authentication was successful. • `false`: the authentication failed.	Read/write
API Gateway only `client.third_party.response.user`	The user for third-party authentication.	Read/write
API Gateway only `client.title`	The title for the credentials that are received in the request.	Read/write
DataPower Gateway (v5 compatible) only `env.name`	The name of the Catalog in which the API is published.	Read/write
DataPower Gateway (v5 compatible) only `env.path`	The path segment that represents this Catalog.	Read/write
API Gateway only `error.message`	The message text for the error.	Read/write
API Gateway only `error.name`	The name of the error.	Read-only
API Gateway only `log`	The transaction data gathered in an assembly log action.	Read/write
`message.attachments`	The array of attachments in a multipart message.	Read/write
`message.attachments[].body`	The attachment payloads in a multipart message.	Read/write
`message.attachments[].headers`	The attachment headers in a multipart message.	Read/write
`message.body`	The payload of the request or response message. Note: The `message.body` context variable is not supported with `getContext()` function. Use the `getvariable()` function instead.	Read/write
`message.headers.`*`name`*	The value of the current named header of the message or of the current named header of the root part of a multipart message. The *name* segment is case-insensitive.	Read/write
API Gateway only `message.original.headers.`*`name`*	The value of the original named header of the HTTP message. When the assembly contains an invoke action, the value is automatically populated by the original named header of the response message from the invoked URL. The *name* segment is case-insensitive.	Read-only
API Gateway only `message.package.headers.`*`name`*	The value of the current named header of the multipart message. The *name* segment is case-insensitive.	Read/write
`message.status.code`	The HTTP status code of the response.	Read/write
`message.status.reason`	The HTTP reason phrase of the response.	Read/write
API Gateway only `message.variables.`*`name`*	The value of a property in the message. For example, when `myvar` is the property in the message, you can retrieve the value of the `myvar` property by reading `message.variables.myvar`.	Read/write
`plan.name`	The name of the plan.	Read/write
`plan.id`	The unique identifier of the plan.	Read/write
`plan.rate-limit`	The rate limit of the plan, which is the number of API calls per time interval.	Read/write
API Gateway only `plan.rate-limit[index].hard-limit`	The hard limit setting for the rate limit scheme that is identified by `[index]`.	Read/write
API Gateway only `plan.rate-limit[index].value`	The value of the rate limit scheme that is identified by `[index]`. The syntax is `rate/interval`.	Read/write
`plan.spaceId`	The unique identifier of the space that the plan is contained within.	Read/write
`plan.spaceName`	The name of the space that the plan is contained within.	Read/write
`plan.version`	The version number of the plan.	Read/write
`request.authorization`	The parsed HTTP `authorization` header.	Read-only
`request.body`	The payload from the incoming request.	Read-only
`request.content-type`	Normalized content-type value.	Read-only
`request.date`	A date object that represents approximately when the request was received by the Gateway.	Read-only
`request.headers.`*`name`*	The value of the original named header of the HTTP request, or the value of the current named header of the root part of a multipart request. The *name* segment is case-insensitive. The request header can be modified only in preflow policies. For more information, see Customizing the preflow policies.	Read/write

Name	Description	Permission
API Gateway only **request.original.headers.*name***	The value of the original named header of the HTTP request. This variable is created only when the request header is modified. The *name* segment is case-insensitive.	Read-only
API Gateway only **request.package.headers.*name***	The value of the named header of the multipart request. The *name* segment is case-insensitive.	Read-only
API Gateway only **request.parameters**	You can obtain your incoming parameters from path and query parameters.	
API Gateway only **request.parameters.*name*.locations**	An array of strings that specify the parameter types associated with the parameter specified in name. The supported keywords for parameter types are as follows. **formData** The parameter is in the body as form data **header** The parameter is in the request header **path** The parameter is in the path **query** The parameter is in the query	Read-only
API Gateway only **request.parameters.*name*.values**	An array containing values of the parameter types associated with the parameter specified in name. For example, when the first value is **path** in **request.parameters.myparam.locations[]**, the first value in **request.parameters.myparam.values[]** is the array of path values associated with **myparam**. Request parameters can be modified only in preflow policies. For more information, see Customizing the preflow policies.	Read/write
API Gateway only **request.original.parameters.*name*.values**	An array containing the original values of the parameter types associated with the parameter specified in *name*. This variable is created only when a request parameter value is modified.	Read-only
request.path	The path section of the **request.uri** that starts with the base path of the API, including the '/' character that begins the base path.	Read-only
API Gateway only **request.protocol**	The protocol that is used to receive the request: **http** or **https**.	Read-only
request.querystring	The request query string without the leading question mark.	Read-only
request.search	The request query string with the leading question mark.	Read-only
request.uri	The full HTTP request URI from the application.	Read-only
request.verb	The HTTP verb of this request.	Read-only
API Gateway only **session.apiGateway**	The gateway that receives the request.	Read-only
API Gateway only **session.apiGatewayName**	The name of the API gateway as defined in the API Manager.	Read-only
API Gateway only **session.clientAddress**	The address of the client that sent the request.	Read-only
API Gateway only **session.domainName**	The name of the domain that the gateway belongs to.	Read-only
API Gateway only **session.globalTransactionID**	The global transaction ID in the logs.	Read-only
API Gateway only **session.localAddress**	The address of the gateway on the DataPower® Gateway.	Read-only
API Gateway only **session.timeStarted**	The time that the gateway started to process the request.	Read-only
API Gateway only **session.transactionID**	The transaction ID of the gateway request.	Read-only
system.datetime	Returns a string that represents the current date and time in the system time zone of the gateway.	Read-only
system.time	Returns a string that represents the current time in the system time zone of the gateway.	Read-only
system.time.hour	Returns a number 0 - 23 inclusive, representing the hour of the current time in the system time zone of the gateway.	Read-only
system.time.minute	Returns a number 0 - 59 inclusive representing the minute of the current time in the system time zone of the gateway.	Read-only
system.time.seconds	Returns a number 0 - 59 inclusive representing the seconds of the current time in the system time zone of the gateway.	Read-only
system.date	Returns a string that represents the current date in the system time zone of the gateway.	Read-only
system.date.day-of-week	Returns a number 1 - 7 (Monday to Sunday) inclusive representing the day of the week in the system time zone of the gateway.	Read-only
system.date.day-of-month	Returns a number 1 - 31 representing the day of the month in the system time zone of the gateway.	Read-only
system.date.month	Returns a number 1 - 12 representing the month in the system time zone of the gateway.	Read-only
system.date.year	Returns a four-digit number that represents the year in the system time zone of the gateway.	Read-only
system.timezone	Returns a system time zone ISO 8601 identifier for the gateway, which might include a sign, a two-digit hour, and minutes. For example, **-04:00**.	Read-only

OAuth context variables - DataPower Gateway (v5 compatible)

The OAuth context variables described in this section apply only to the DataPower Gateway (v5 compatible). For details of the OAuth context variables that apply to the DataPower API Gateway, see OAuth context variables.

Table 2. OAuth context variables (DataPower Gateway (v5 compatible)).
Note: Most OAuth context variables are available only when IBM API Connect is acting as the OAuth resource server. However, the `oauth.introspect` variables are also available when integrating with third party providers.

Name	Description
`oauth.access-token`	If the request is authenticated with OAuth, this variable contains the access token string.
`oauth.miscinfo`	This variable contains information explicitly included in the Authentication URL and Metadata URL headers. For more information, see Authenticate URL.
`oauth.not-after`	If the request is authenticated with OAuth, this variable contains the date when the token expires.
`oauth.not-before`	If the request is authenticated with OAuth, this variable contains the date when the token was issued.
`oauth.resource-owner`	If the request is authenticated with OAuth, this variable contains the name of the resource owner.
`oauth.scope`	If the request is authenticated with OAuth, this variable contains the scope of this access token.
`oauth.introspect.active`	Always available to introspection. Boolean value.
`oauth.introspect.response`	Always available to introspection. Shows the complete current response payload. Example payload value: `{"active":true, "client_id", "xxx-xxx", "token_type", "bearer", "scope":"neon"}`

Other variables might be available from the third party, in the form of: `oauth.introspect.<variable>`	Decoding the previous example payload, the following variables are made available for further processing.
	
	`oauth.introspect.client_id: xxx-xxx`
	`oauth.introspect.token_type: bearer`
	`oauth.introspect.scope: neon`

Application certificate context variables

The following table describes context variables that are available when a certificate is used to verify access to an API, although these will vary depending on the signature mechanism that is being used; for more information, see the Internet X.509 Public Key Infrastructure Certificate and CRL Profile specification.

Table 3. Application certificate context variables

Name	Description
`application.certificate.Base64`	Base64 format.
`application.certificate.fingerprint`	Fingerprint
`application.certificate.Version`	Version
`application.certificate.SerialNumber`	Serial number
`application.certificate.SignatureAlgorithm`	Signing algorithm
`application.certificate.Issuer`	The issuer of the certificate
`application.certificate.Subject`	Subject
`application.certificate.NotBefore`	Not valid before this date
`application.certificate.NotAfter`	Not valid after this date
`application.certificate.SubjectPublicKeyAlgorithm`	Algorithm for the subject public key
`application.certificate.SubjectPublicKeyBitLength`	Length for the subject public key
`application.certificate.KeyValue.type`	Various context variables that depend on the algorithm and key. The following are possible context variables: • `application.certificate.KeyValue.RSAKeyValue.Modulus` • `application.certificate.KeyValue.RSAKeyValue.Exponent`

API activity logging context variables

If activity logging is enabled for an API, a `log` context variable is created; the `log` context variable contains the data relating to an API execution event. On completion of API execution, the `log` context variable is written to an API event record that is stored for subsequent access by API analytics. For details of the fields contained in the `log` context variable, see API event record fields.

The way in which you enable and configure activity logging depends on the type of gateway you are using, as follows:

- If you are using the DataPower API Gateway, you configure activity logging in the API configuration settings; see Configuring activity logging (OpenAPI 2.0) or Configuring activity logging (OpenAPI 3.0).
- If you are using the DataPower Gateway (v5 compatible), you configure activity logging by adding an Activity Log policy to the API assembly.

The activity logging configuration defines the default content of the `log` context variable, but you can modify it in an API assembly; for example:

- Add your own data values by using a Set Variable policy.
- Remove or redact data values by using a Redaction policy; see Redaction - DataPower API Gateway or Redaction - DataPower Gateway (v5 compatible).
- API Gateway only Modify or replace the `log` context variable by using a Log policy.

GraphQL context variables

The GraphQL query and response variables generated from API processing and assembly actions are stored in the API context in the `message.attributes.graphql` context variables.

Table 4. GraphQL context variables

Name	Description

Name	Description
`message.attributes.graphql`	The query and response attributes of the GraphQL message.
`message.attributes.graphql.query`	The attributes of a GraphQL query, including the query string, operation name, and variables.
`message.attributes.graphql.query.query`	The GraphQL query string.
`message.attributes.graphql.query.variables`	The map of GraphQL variables.
`message.attributes.graphql.query.operationName`	The name of the operation to execute.
`message.attributes.graphql.query.operationType`	The type of the operation to execute.
`message.attributes.graphql.request`	The attributes of the GraphQL request.
`message.attributes.graphql.request.fieldCost`	The weighted sum of the cost of all fields accessed in the query. The cost of each field access is configured in the schema. This count includes fields whose field costs are not set to 0.0.
`message.attributes.graphql.request.fieldCounts`	The maximum number of times that each field can be retrieved by the given query. This number is equal to the number of times that the resolver responsible for producing the value of the field is required to run. This count includes fields whose field costs are not set to 0.0.
`message.attributes.graphql.request.typeCost`	The weighted sum of the cost of all types retrieved in the query. The cost of each type is configured in the schema. This count includes types whose type costs are not set to 0.0.
`message.attributes.graphql.request.typeCounts`	The maximum number of times that a value of each type can be retrieved by the given query. This count includes types whose type costs are not set to 0.0.
`message.attributes.graphql.response.fieldCost`	The weighted sum of the cost of all fields in the response to the query. The cost of each field access is configured in the schema. If this sum cannot be calculated based on the analysis configuration in the request, the field cost is expressed as the maximum integer value that can be assigned.
`message.attributes.graphql.response.fieldCounts`	The number of times that each field was retrieved in the response to the query. This number is equal to the number of times that the resolver responsible for producing the value of the field is required to run. If this sum cannot be calculated based on the analysis configuration in the request, the field count is expressed as the maximum integer value that can be assigned.
`message.attributes.graphql.response.typeCost`	The weighted sum of the cost of all types in the response to the query. The cost of each type is configured in the schema. If this sum cannot be calculated based on the analysis configuration in the request, the type cost is expressed as the maximum integer value that can be assigned.
`message.attributes.graphql.response.typeCounts`	The number of times that a value of each type was contained in the response to the query. If this sum cannot be calculated based on the analysis configuration in the request, the type count is expressed as the maximum integer value that can be assigned.
`message.attributes.parse.GraphQLIntrospection`	The introspection type of the assembly GraphQL introspect action. When a GraphQL query is parsed, this context variable is populated with one of the following values: • `not-introspection` • `custom-introspection` • `default-introspection`

Related concepts

- Variable references in API Connect

API Gateway only

OAuth context variables

You can customize the OAuth security flow in a native OAuth provider by adding multiple OAuth policies to the assembly. Each OAuth policy takes input from context variables and writes output to context variables. By using these OAuth context variables you can manipulate the original request, and process the output from an OAuth policy.

Note: The OAuth context variables described here apply only to the DataPower® API Gateway. For details of the OAuth context variables that apply to the DataPower Gateway (v5 compatible), see OAuth context variables - DataPower Gateway (v5 compatible).
There are two main types of OAuth context variables:

- Information variables, outputs of an OAuth policy.
- Processing variables, processed during the execution of an OAuth policy.

Information variables

These context variables are outputs of an OAuth policy. They are not used in the processing of the OAuth action.

`oauth.result`
: The result of the latest action; returns `SUCCESS` or `FAILURE`.

`oauth.settings.variable_name`
: Basic settings from the OAuth provider. The following context variables are available:

  `oauth.settings.allowed_scopes`
  `oauth.settings.access_token_ttl`

`oauth.settings.authorization_code_ttl`
`oauth.settings.refresh_token_ttl`
`oauth.settings.refresh_token_limit`
`oauth.settings.maximum_consent_ttl`

`oauth.executed_components[0].`***`variable_name`***
The components that have executed in this transaction, with their result. If a failure occurs then the error and error description will be added. The following context variables are available:

`oauth.executed_components[0].result`
`oauth.executed_components[0].type`
`oauth.executed_components[0].error_description`
`oauth.executed_components[0].error`

The possible values for the `type` variable are as follows:

`ValidateRequestComponent`
`GenerateAZCodeComponent`
`VerifyAZCodeComponent`
`VerifyRefreshTokenComponent`
`GenerateAccessTokenComponent`
`IntrospectTokenComponent`
`RevokeTokenComponent`
`CollectMetaDataComponent`

The following example shows a corresponding JSON extract from the OAuth context:

```
"executed_components": [
  {
    "type": "ValidateRequestComponent",
    "result": "SUCCESS"
  },
  {
    "type": "GenerateAccessTokenComponent",
    "result": "SUCCESS"
  }
]
```

`oauth.verified_access_token.`***`variable_name`***
The token that was verified in the security requirement for a native OAuth provider. The following context variables are available:

`oauth.verified_access_token.access_token`

`oauth.verified_access_token.client_id`
`oauth.verified_access_token.consented_on`
`oauth.verified_access_token.consented_on_text`
`oauth.verified_access_token.grant_type`
`oauth.verified_access_token.misc_info`
`oauth.verified_access_token.not_after`
`oauth.verified_access_token.not_after_text`
`oauth.verified_access_token.not_before`
`oauth.verified_access_token.not_before_text`
`oauth.verified_access_token.one_time_use`
`oauth.verified_access_token.resource_owner`
`oauth.verified_access_token.scope`

`oauth.third_party.`***`variable_name`***
The token that was verified in the security requirement for a third party OAuth provider. Provides the following information:

- The response headers from the external, third-party OAuth provider.
- The body of response from the external, third-party OAuth provider.
- Whether the response is cached. When cached, the value is `true`. When not cached, the value is `false`.

The following context variables are available:

`oauth.third_party.headers`
`oauth.third_party.response`
`oauth.third_party.cached`

`oauth.code.`***`variable_name`***
The code that was generated during the execution of a component of type `GenerateAZCodeComponent`. The following context variables are available:

`oauth.code.client_id`
`oauth.code.code`
`oauth.code.redirect_uri`
`oauth.code.resource_owner`
`oauth.code.scope`

`oauth.token.`***`variable_name`***
The token that was generated during the execution of a component of type `GenerateAccessTokenComponent`. The following context variables are available:

`oauth.token.token_type`
`oauth.token.access_token`
`oauth.token.scope`
`oauth.token.expires_in`
`oauth.token.consented_on`
`oauth.token.redirect_uri`
`oauth.token.resource_owner`
`oauth.token.client_id`
`oauth.token.refresh_token`
`oauth.token.refresh_token_expires_in`
`oauth.token.refresh_token_count`

`oauth.introspect.`***`variable_name`***
The token that was introspected during execution of a component of type `IntrospectTokenComponent`. The following context variables are available:

`oauth.introspect.active`
`oauth.introspect.scope`
`oauth.introspect.client_id`
`oauth.introspect.resource_owner`
`oauth.introspect.token_type`
`oauth.introspect.grant_type`
`oauth.introspect.ttl`
`oauth.introspect.expires`
`oauth.introspect.expires_text`
`oauth.introspect.iat`
`oauth.introspect.not_before`
`oauth.introspect.not_before_text`
`oauth.introspect.consented_on`
`oauth.introspect.consented_on_text`
`oauth.introspect.one_time_use`

`oauth.external_manager.`*`variable_name`*
Token management when using an external token manager. Provides the following information:

- The response headers from the external management server.
- The response JSON body from the external management server.
- When caching is enabled, whether the response is found in the cache or set in the cache. When caching is not enabled, that the response is not set in the cache.

The following context variables are available:

`oauth.external_manager.headers`
`oauth.external_manager.response`
`oauth.external_manager.cached`

Processing variables

These context variables are processed during the execution of an OAuth policy in your API assembly. If they are set prior to any OAuth policy then the value overrides what was sent in the request. They can also be modified between OAuth policies to manipulate the next component that executes. The following context variables are available:

```
oauth.processing.assertion
oauth.processing.client_id
oauth.processing.client_secret
oauth.processing.grant_type
oauth.processing.redirect_uri
oauth.processing.scope
oauth.processing.response_type
oauth.processing.state
oauth.processing.resource_owner
oauth.processing.refresh_token
oauth.processing.code
oauth.processing.token
oauth.processing.token_type_hint
oauth.processing.nonce
oauth.processing.max_age
oauth.processing.oidc_values_requested
oauth.processing.id_token_requested
oauth.processing.oidc_signing_algorithm
oauth.processing.code_challenge
oauth.processing.code_challenge_method
oauth.processing.code_verifier

oauth.processing.metadata.access_token
oauth.processing.metadata.payload
oauth.processing.metadata.azcode_miscinfo

oauth.processing.verified_code.client_id
oauth.processing.verified_code.resource_owner
oauth.processing.verified_code.misc_info
oauth.processing.verified_code.scope
oauth.processing.verified_code.is_verified
oauth.processing.verified_code.nonce

oauth.processing.verified_refresh_token.client_id
oauth.processing.verified_refresh_token.resource_owner
oauth.processing.verified_refresh_token.misc_info
oauth.processing.verified_refresh_token.scope
oauth.processing.verified_refresh_token.refresh_token_count
oauth.processing.verified_refresh_token.is_verified
oauth.processing.verified_refresh_token.one_time_use
oauth.processing.verified_refresh_token.grant_type
```

The following example shows the OpenAPI source code for a **`gateway-script`** policy that executes before an OAuth policy in your API assembly and adds a custom scope to the request:

```
// Add another custom scope to the request
let scope = context.get("request.parameters.scope.values[0]");
if (scope)
  context.set("oauth.processing.scope", scope + " custom");
```

The following example shows the OpenAPI source code for a **`gateway-script`** policy that between OAuth policies in your assembly and modifies the scope depending on the resource owner:

```
// Check resource owner and modify the scope
let owner = context.get("oauth.processing.resource_owner");
let scope = context.get("oauth.processing.scope");

if (owner === 'admin') {
```

```
    context.set("oauth.processing.scope", scope + " admin");
} else {
    context.set("oauth.processing.scope", scope + " customer");
}
```

Advanced scope context variables

The following context variables are used in OAuth advanced scope checking. They are populated by the values entered in the Endpoint and TLS Client Profile fields for the Application scope check and Owner scope check in the native OAuth provider configuration. Do not change these values as this might result in incorrect advanced scope operation.

For information on scope configuration, see Configuring scopes for a native OAuth provider. For information on advanced scope checking, see Scope.

```
oauth.advscope.app.url
oauth.advscope.app.tls-profile
oauth.advscope.own.url
oauth.advscope.own.tls-profile
```

Custom error context variables

These context variables are used to stop OAuth processing and specify the respective error details. They can be used before OAuth processing takes place.

```
oauth.custom_error.status.code
oauth.custom_error.status.reason
oauth.custom_error.message
oauth.custom_error.description
```

Related reference

- Example - using multiple OAuth policies in an OAuth provider assembly

Error cases supported by assembly catches

Several error cases that can be returned by the assembly are available to the catch function.

ConnectionError
: An error occurred while establishing a connection to another URL.

JavaScriptError
: An error occurred while executing JavaScript or GatewayScript in a policy.

PropertyError
: An error occurred due to an incorrect property during an invoke call or during the execution of a set-variable policy when an action was not set, add, or clear.

RedactionError
: An error occurred during the redaction of a field as part of a redact policy.

TransformError
: An error occurred during a transformation policy.

RuntimeError
: An otherwise unspecified error occurred.

BadRequestError
: An error occurred while trying to access the request. This is not supported by the default catch; to catch this condition, specify it in your catch definition.

SOAPError
: A SOAP Fault response was received as the result of a web service invoke request.

OperationError
: An invoke request received a response with an unsuccessful HTTP status code.

ValidateError
: An error occurred while trying to schema validate a message payload.

AssemblyRateLimitError
: The request exceeds the transaction count threshold specified in the API's Plan.

UnauthorizedError
: The API cannot be invoked based on the client ID and/or client secret provided, or id/secret were specified in the wrong location. This is not supported by the default catch; to catch this condition, specify it in your catch definition.

ForbiddenError
: The application making the API request has been disabled or is not active. This is not supported by the default catch; to catch this condition, specify it in your catch definition.

API and Product definition template examples

You can use template files when creating API and Product definitions. Template files are Handlebars templates containing variables of the form **{{variable-name}}** that are substituted with values when you create the API or Product definition. This topic provides the default Handlebar templates used by **apic** to create Products and APIs as examples you can use to copy and customize for your own use.

Default API definition template (OpenAPI 2.0)

The following example template, which is for an OpenAPI 2.0 API definition, is the default template that developer toolkit uses when you create an API definition. Copy this example template into your own template file (which must have the **.hbs** extension), then edit it for your purposes. Template variables are enclosed by double curly

braces, for example `{{name}}`. For information on template variables, see Template variables for API and Product definitions. For more information on Handlebars, see https://handlebarsjs.com/.

```
swagger: '2.0'

info:
  x-ibm-name: {{name}}
  title: {{title}}
  version: {{version}}

schemes:
{{#if schemes}}
  {{#each schemes}}
    - {{this}}
  {{/each}}
{{else}}
    - https
{{/if}}
host: {{hostname}}
basePath: {{basepath}}

consumes:
  - application/json
produces:
  - application/json

securityDefinitions:
 clientIdHeader:
   type: apiKey
   in: header
   name: X-IBM-Client-Id
 clientSecretHeader:
   in: "header"
   name: "X-IBM-Client-Secret"
   type: "apiKey"

security:
 -
   clientIdHeader: []
   clientSecretHeader: []

x-ibm-configuration:
  testable: true
  enforced: true
  gateway: datapower-gateway
  cors:
    enabled: true
  catalogs:
    apic-dev:
      properties:
        runtime-url: $(TARGET_URL)
    sb:
      properties:
        runtime-url: 'http://localhost:4001'
  assembly:
    execute:
      - invoke:
        {{#if targeturl}}
          target-url: {{targeturl}}
        {{else}}
          target-url: $(runtime-url)$(request.path)
        {{/if}}
paths:

  /users:

    post:
      summary: Create a user
      description: Create a new user
      operationId: userCreate
      externalDocs:
        description: Blah
        url: http://host/docs-about-routes-post
      tags:
        - Users
      responses:
        '201':
          description: 'Success'
          schema:
            $ref: '#/definitions/User'
        default:
          description: 'Unexpected error'
          schema:
            $ref: '#/definitions/Error'

    get:
      summary: User list
      description: Get a list of users
      operationId: userList
      externalDocs:
        description: Blah
        url: http://host/docs-about-routes-post
      tags:
        - Users
      responses:
```

```
        '200':
          description: 'Success'
          schema:
            $ref: '#/definitions/UserList'
        default:
          description: 'Unexpected error'
          schema:
            $ref: '#/definitions/Error'

  /users/{user}:

    get:
      summary: Retrieve the User
      description: Retrieve the User
      operationId: userGet
      tags:
        - Users
      parameters:
        - name: user
          in: path
          description: User id or name
          required: true
          type: string
      responses:
        '200':
          description: 'Success'
          schema:
            $ref: '#/definitions/User'
        default:
          description: 'Unexpected error'
          schema:
            $ref: '#/definitions/Error'

    patch:
      summary: Update User
      description: Update User
      operationId: userUpdate
      tags:
        - Users
      parameters:
        - name: user
          in: path
          description: User id or name
          required: true
          type: string
        - name: payload
          in: body
          description: User to update
          required: true
          schema:
            $ref: '#/definitions/UserUpdate'
      responses:
        '200':
          description: 'Success'
          schema:
            $ref: '#/definitions/User'
        default:
          description: 'Unexpected error'
          schema:
            $ref: '#/definitions/Error'

    delete:
      summary: Delete the User
      description: Delete the User
      operationId: userDelete
      tags:
        - Users
      parameters:
        - name: user
          in: path
          description: User id or name
          required: true
          type: string
      responses:
        '204':
          description: 'Successful delete'
        default:
          description: 'Unexpected error'
          schema:
            $ref: '#/definitions/Error'

definitions:

  User:
    type: object
    additionalProperties: false

  UserUpdate:
    type: object
    additionalProperties: false

  UserList:
    type: object
    additionalProperties: false

  Error:
```

```
    type: object
    additionalProperties: false
    properties:
      status:
        type: integer
      message:
        type:
          - string
          - array

tags:
  - name: Users
    description: Tags on all the user operations
    externalDocs:
      description: External information about Users
      url: http://host/url-of-my-entire-set-of-tag-docs-for-this-tag
  - name: Routes
    description: Tags on all the route operations
    externalDocs:
      description: External information about Routes
      url: http://host/url-of-my-entire-set-of-tag-docs-for-this-tag
```

OpenAPI 3.0 API definition template

The following example template is for an OpenAPI 3.0 API definition. Copy this example template into your own template file (which must have the **.hbs** extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example **{{name}}**. For information on template variables, see Template variables for API and Product definitions. For more information on Handlebars, see https://handlebarsjs.com/.

```
openapi: '3.0.3'

info:
  x-ibm-name: {{name}}
  version: {{version}}
  title: {{title}}

x-ibm-configuration:
  assembly:
    execute:
    - invoke:
        title: invoke
        version: 2.0.0
        target-url: 'http://apicsrv01.dp.rtp.raleigh.ibm.com/sample.json'
  cors:
    enabled: true
  enforced: true
  phase: realized
  testable: true
  gateway: datapower-api-gateway

servers:
 - url: "/oauth-supported-flows"

paths:
  "/getData":
    get:
      summary: Operation Get Branches
      operationId: basicImplicit
      responses:
        default:
          description: Default response

security:
  - Oauth:
    - scopePub

components:
  securitySchemes:
    clientIdHeader:
      type: apiKey
      in: header
      name: clientHeader
    clientSecretHeader:
      type: apiKey
      in: header
      name: X-IBM-Client-Secret
    client_id:
      type: apiKey
      in: query
      name: client_id
    client_secret:
      type: apiKey
      in: query
      name: client_secret
```

OAuth 2.0 API definition template

The following example template is the default template that developer toolkit uses when you create an OAuth 2.0 API definition with the command **apic create:api --template oauth**. Copy this example template into your own template file (which must have the **.hbs** extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example **{{name}}**. For information on template variables, see Template variables for API and Product definitions. For more information on Handlebars, see https://handlebarsjs.com/.

```
swagger: "2.0"

info:
  x-ibm-name: {{name}}
  title: {{title}}
  version: {{version}}

schemes:
{{#if schemes}}
    {{#each schemes}}
        - {{this}}
    {{/each}}
{{else}}
    - https
{{/if}}
host: {{hostname}}
basePath: {{basepath}}

securityDefinitions:
  clientID:
    description: "application's client_id"
    in: "query"
    name: "client_id"
    type: "apiKey"

security:
- clientID: []

paths:

  /oauth2/authorize:

    get:
      produces:
        - text/html
      summary: endpoint for Authorization Code and Implicit grants
      description: description
      parameters:
        - name: response_type
          in: query
          description: request an authorization code or or access token (implicit)
          required: true
          type: string
          enum:
            - code
            - token
        - name: client_id
          in: query
          description: Application client ID
          required: true
          type: string
        - name: scope
          in: query
          description: Scope being requested
          type: string
          required: true
        - name: redirect_uri
          in: query
          type: string
          description: URI where user is redirected to after authorization
          required: false
        - name: state
          in: query
          type: string
          description: This string will be echoed back to application when user is redirected
          required: false
      responses:
        302:
          description: |
            Redirect to the clients redirect_uri containing one of the following
            - **authorization code** for Authorization code grant
            - **access token** for Implicity grant
            - **error** in case of errors, such as the user has denied the request
        200:
          description: An HTML form for authentication or authorization of this request.
      security:
      - clientID: []

    post:
      consumes:
        - application/x-www-form-urlencoded
      produces:
        - text/html
      summary: submit approval to authorization code or access token
      description: |
        Submit resource owners approval (or rejection) for the OAuth2 Server to issue an
        authorization code or access token to the application.
      security:
      - clientID: []
      parameters:
        - name: client_id
          in: formData
          description: application requesting the access code or token
          required: true
          type: string
        - name: scope
```

```
        in: formData
        description: requested scope of this authorization
        required: true
        type: string
      - name: resource-owner
        in: formData
        description: resource owners user name
        required: true
        type: string
      - name: redirect_uri
        in: formData
        description: URI the application is requesting this code or token to be redirected to
        required: true
        type: string
      - name: original-url
        in: formData
        description: URL of the original authorization request
        required: true
        type: string
      - name: dp-state
        in: formData
        description: state information provided in the authorization form
        required: true
        type: string
      - name: dp-data
        in: formData
        description: state information provided in the authorization form
        required: true
        type: string
      #- name: response_type
      #  in: formData
      #  description:
      #  required: true
      #  type: string
    responses:
      200:
        description: Cool

  /oauth2/token:

    post:
      consumes:
        - application/x-www-form-urlencoded
      produces:
        - application/json
      summary: Request Access Tokens
      description: |
        This endpoint allows requesting an access token following one of the flows below:
        - Authorization Code (exchange code for access token)
        - Client Credentials (2-legged, there isnt resource owner information)
        - Resource Owner Password Credentials (2-legged, client provides resource owner name and password)
        - Refresh Token (exchange refresh token for a new access code)

        The table below indicates the required parameters for each specific grant_type options.
        Empty cells indicate a parameter is ignored for that specific grant type.

        Client authentication:
        - Confidential clients should authenticate using HTTP Basic Authentication. Alternatively, they may post
          their client_id and client_secret information as a formData parameter.
        - Public clients should send their client_id as formData parameter.

        | grant_type           | code       | client_credentials | password    | refresh_token |
        |----------------------|------------|--------------------|-------------|---------------|
        | client_id            | required*  | required*          | required*   | required*     |
        | client_secret        | required*  | required*          | required*   | required*     |
        | code                 | required   |                    |             |               |
        | redirect_uri         | required   |                    |             |               |
        | username             |            |                    | required    |               |
        | password             |            |                    | required    |               |
        | scope                |            | optional           | optional    |               |
        | refresh_token        |            |                    |             | required      |

        The implicit grant requests, see /oauth2/authorize.

      security: []

      parameters:
        - name: grant_type
          in: formData
          description: Type of grant
          type: string
          required: true
          enum:
            - authorization_code
            - password
            - client_credentials
            - refresh_token
        - name: client_id
          in: formData
          description: Application client ID, can be provided in formData or using HTTP Basic Authentication
          required: false
          type: string
        - name: client_secret
          in: formData
          description: Application secret, must be provided in formData or using HTTP Basic Authentication
          required: false
```

```
          type: string
        - name: code
          in: formData
          description: Authorization code provided by the /oauth2/authorize endpoint
          required: false
          type: string
        - name: redirect_uri
          in: formData
          description: required only if the redirect_uri parameter was included in the authorization request /oauth2/authorize;
their values MUST be identical.
          required: false
          type: string
        - name: username
          in: formData
          type: string
          description: Resource owner username
          required: false
        - name: password
          in: formData
          type: string
          description: Resource owner password
          required: false
        - name: scope
          in: formData
          type: string
          description: Scope being requested
          required: false
        - name: refresh_token
          in: formData
          type: string
          description: The refresh token that the client wants to exchange for a new access token (refresh_token grant_type)
          required: false
      responses:
        200:
          description: json document containing token, etc.
          schema:
            $ref: "#/definitions/access_token_response"
        400:
          description: json document that may contain additional details about the failure

x-ibm-configuration:
  testable: true
  enforced: true
  phase: "realized"
  oauth2:
    client-type: public #or confidential
    scopes:
      scope1: Description 1
      scope2: Description 2
      scope3: Description 3
    grants:
      - application
      - password
      - accessCode
      - implicit

    identity-extraction:
      type: default-form  #If identity extraction is not there use this form.
      #type: basic
      #type: custom-form  #Customer provided form (needs location)
      #type: redirect     #Redirects user to authenticate somewhere else
      #custom-form:
      #  url: https://example.com/authentication/form
      #  tls-profile: tls-profile-1
      #redirect-url: https://example.com/external/form

    authentication:
      x-ibm-authentication-url:
        url: https://example.com/auth/url
        tls-profile: tls-profile-4
      #x-ibm-authentication-registry: ldap-1

    authorization:
      type: authenticated #If the authorization section is missing this is the default
      #type: default-form
      #type: custom-form
      #custom-form:
      #  url: https://example.com/authorization/form
      #  tls-profile: tls-profile-2

    refresh-token:
      count: 2048 # If this section is missing default is 0
    revocation:
      url: ""
      tls-profile: ""

definitions:

  access_token_response:
    type: object
    additionalProperties: false
    required:
      - token_type
      - access_token
      - expires_in
    properties:
```

```
      token_type:
        enum:
          - bearer
      access_token:
        type: string
      expires_in:
        type: integer
      scope:
        type: string
      refresh_token:
        type: string
```

Product definition template

The following example template is the default template that developer toolkit uses when you create a Product definition. Copy this example template into your own template file (which must have the **.hbs** extension), then edit it for your purposes. Template variables are enclosed by double curly braces, for example **{{name}}**. For information on template variables, see Template variables for API and Product definitions. For more information on Handlebars, see https://handlebarsjs.com/.

```
product: '1.0.0'

info:
  name: {{name}}
  title: {{title}}
  version: {{version}}

{{#isEmpty apis}}
{{else}}
apis:
{{/isEmpty}}
{{#each apis}}
  '{{@key}}':
    $ref: {{this}}
{{/each}}

visibility:
  view:
    type: public
  subscribe:
    type: authenticated

plans:
  default:
    title: Default Plan
    description: Default Plan
    approval: false
    rate-limit:
      value: 100/hour
      hard-limit: false
```

Related tasks

- Creating and using API and Product definitions templates

Related reference

- Template variables for API and Product definitions

Template variables for API and Product definitions

You can use template files when creating OpenAPI 2.0 and OpenAPI 3.0 APIs, and when creating Product definitions. Template files are Handlebars templates containing variables of the form **{{variable-name}}** that are substituted with values when you create the API or Product definition.

Product definition variables

The following table describes the Handlebars template variables that can be used in a product definition file. For more information on Handlebars, see https://handlebarsjs.com/. Product definition template files must have a **.hbs** filename extension.

Table 1. Product definition Handlebars template variables

Variable	Type	Description

{{apis}}	Array of string	The APIs to which the product definition refers. After substitution, the array values become the values of **apis.routes[*n*].$ref** fields, for example:
		`apis:`
		`  'routes':`
		`  $ref: apidef.yaml`
		`  ...`

{{name}}	String	Value of **info.x-ibm-name** field.
{{title}}	String	Value of **info.title** field.
{{version}}	String	Value of **info.version** field.

API definition variables

The following table describes the Handlebars template variables that can be used in an API definition file. For more information on Handlebars, see https://handlebarsjs.com/. API definition template files must have a `.hbs` filename extension.

Table 2. API definition Handlebars template variables

Variable	Type	Description
{{basepath}}	String	Base path on which the API is served, which is relative to the `host`.
{{definitions}}	OpenAPI definitions objectconverted to a YAML string ("stringified").	For a LoopBack API, contains data types that can be consumed and produced by operations. These data types can be primitives, arrays or models.
{{definitionsObj}}	OpenAPI definitions objectconverted to a YAML string ("stringified").	For a LoopBack API, contains data types that can be consumed and produced by operations. These data types can be primitives, arrays or models.
{{hostname}}	String	Value of the `host` field.
{{name}}	String	Value of `info.x-ibm-name` field.
{{paths}}	OpenAPI paths object	For a LoopBack API, contains the relative paths to the individual endpoints. The path is appended to `{{basePath}}` to construct the full URL.
{{pathsObj}}	OpenAPI paths object	For a LoopBack API, contains the relative paths to the individual endpoints. The path is appended to `{{basePath}}` to construct the full URL.
{{schemes}}	Array of string	Transfer protocol of the API. Values must be one of: `"http"`,`"https"`, `"ws"`, or `"wss"`.
{{targeturl}}	String	Value of `x-ibm-configuration.assembly.execute[invoke]`. Default is `$(runtime-url)$(request.path)`.
{{title}}	String	Value of `info.title` field.
{{version}}	String	Value of `info.version` field.

Related tasks

- Creating and using API and Product definitions templates

Related reference

- API and Product definition template examples

Managing your APIs

You manage your APIs by using the API Manager user interface of IBM® API Connect. You can also analyze your API usage by using the analytics that are provided and socialize your APIs in a developer portal.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

Why use APIs?

Whether you are a business user, an IT user, or an application developer, APIs are increasingly important to your business. You can use an API to publicize your company. The assets, data, or services of your company can be provided to external application developers to expand your enterprise and open new markets.

The API Manager UI provides a solution for companies to manage APIs for private internal APIs, and public external APIs. This on-premises offering provides the capabilities that are required so that you can externalize and manage your services as REST or SOAP APIs.

What do you need to know?

Depending on your role, you can complete different tasks relating to managing your API Catalogs and these are outlined in the documentation. Each task is covered in the order that they are executed.

Each task introduces new features of the API Manager UI as they become relevant to the Catalog that you are building.

For security reasons, your session times out after a period of inactivity.

Before you start, check that the browser you are using is supported and meets the required minimum levels; for details, open the Detailed system requirements for a specific product page, search for the IBM API Connect product, then select the required version.

The ways in which you can manage your APIs are described in the following subtopics:

- **API management checklist**
  A summary, with links, of the key tasks for managing APIs in IBM API Connect.
- **Activating your API Manager user account**
  Before you can access the API Manager user interface of IBM API Connect, you must activate the user account that your Cloud Manager administrator invited you to join.
- **Accessing the API Manager user interface**
  The IBM API Connect API Manager user interface provides a range of options for working with your APIs and Products, and managing security.
- **Searching for items in API Manager**
  Use the search feature in IBM API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.

- **Working with Catalogs**
  Products must be staged to a Catalog and then published; application developers are able to access the APIs in a Product by being members of consumer organizations to which the Product is made available. In API Connect, you can create multiple Catalogs. Catalogs are useful for separating Products and APIs for testing before you make them available to consumer organizations. The syndication feature in API Connect means that you can also publish a Product to a Space in a Catalog.
- **Managing the application lifecycle**
  By using the application lifecycle capability in IBM API Connect, the applications that are subscribed to use the API initially have Development status. When application testing is complete, the application developer can request to upgrade the application to Production status.
- **Using syndication in API Connect**
  With the API Connect syndication feature, you can partition your catalogs into *Spaces*. Each space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that space, enabling each team to manage their APIs independently.
- **Working with Products**
  In IBM API Connect, Plans and APIs are grouped together in Products.
- **Working with APIs**
  For a specific Product, you can work with the APIs that are contained in the Plans in that Product.
- **Working with Plans**
  After a Product has been published to a Catalog, you can work with the Plans in that Product.
- **Working with consumer organizations**
  Manage the consumer organizations that access your APIs and Plans when their users sign up to use the IBM API Connect Developer Portal.
- **Working with developer applications**
  You can work with the applications that have been registered in the Developer Portal associated with a Catalog.
- **Working with application subscriptions**
  You can work with the subscriptions to the Plans in all the Products in a specific Catalog. Plan subscriptions are created by application developers in the Developer Portal.
- **Security and authentication**
  In API Manager, you can use TLS profiles to secure the transmission of data between the management server and other API Connect subsystems and external services, and also configure user registries to securely authenticate your Catalogs and APIs.
- **API Analytics**
  You can use API Connect to filter, sort, and aggregate your API event data. You can present the results within correlated charts, tables, and maps to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.
- **Configuring API governance in the API Manager**
  How to add custom API governance rulesets to your API development process, to validate and enforce organizational governance policies and best practices in your provider organization.
- **Administering user access**
  If you have permission to administer users in IBM API Connect, you can add and delete users. After users are removed from an organization through deletion, the user account remains in API Manager.
- **Changing your API Manager password**
  You can change your API Manager password in IBM API Connect.
- **Resolving login problems by increasing HTTP header size**
  You can resolve login problems for the API Manager UI by increasing the maximum HTTP client header size.
- **Reviewing a gateway's processing status**
  Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.
- **Reference**
  Reference information for the API Manager component in IBM API Connect.

Related information

- IBM API Connect overview
- Using the Developer Portal

API management checklist

A summary, with links, of the key tasks for managing APIs in IBM® API Connect.

Task	Description
Activate your API Manager user account	To connect to a Management server using the API Manager user interface, you must be an owner or a member of a provider organization.
Log in to the API Manager user interface	After you have activated your account, you can log in to your Management server from the API Manager user interface.
Configure your Catalogs	APIs are published to a Catalog to make them available to application developers. The URLs for API calls, and for the Developer Portal, are specific to a particular Catalog. By default, a *sandbox* Catalog is provided for API testing, but you can create your own Catalogs; for example, a production Catalog for hosting APIs that are ready for full use.
Manage the membership of your Catalogs	You can add users to your Catalogs, and assign roles and permissions to control their level of access.
Configure API security	You can configure user registries of various types for authenticating API access, TLS profiles for securing data transmission over HTTPS, and OAuth providers for securing third-party website or application access to APIs.
Manage your Products	Move your Products through their lifecycle, from initially staging a Product to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. You can also display analytics information, and control who can see or subscribe to your Products.
Analyze your API usage	You can use IBM API Connect to filter, sort, and aggregate your API event data. You can present the results within correlated charts, tables, and maps to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.

Task	Description
Manage user access to your provider organization	You can add users to your provider organization, remove users, and assign roles and permissions.
Partition your Catalogs into Spaces for syndication	With the syndication feature, you can partition your Catalogs into *Spaces*. Each Space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that Space, enabling each team to manage their APIs independently.

Activating your API Manager user account

Before you can access the API Manager user interface of IBM® API Connect, you must activate the user account that your Cloud Manager administrator invited you to join.

Before you begin

The Cloud Manager administrator invited you to join API Connect .

Procedure

1. Complete the following steps to create your API Manager account:

   If the Identity Provider uses LDAP
   : An invitation email with an activation link is sent. Click the activation link, or paste it in a browser, to log in directly with your user credentials. Upon authentication, the API Connect user record is updated from the backend identity provider.

   If the Identity Provider uses a local registry
   : An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your first name, last name and password. Passwords must have a minimum of 8 characters and contain characters from at least three of the four following categories:
     - Uppercase letters
     - Lowercase letters
     - Numbers
     - Special characters (for example: ! # $ %)

     Note:
     - The email address that you enter on the sign up page must match the email address to which the invitation email was sent, otherwise the account activation fails.
     - If you previously had an account that the administrator removed, and they are inviting you again, you must re-activate your account by using the Sign In option on the page, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail.

   If the Identity Provider uses an authentication URL
   : An invitation email with an activation link is sent. Click the activation link or paste it in a browser. The activation link takes you to a sign up page where you enter your credentials, which then become your user name and password. Upon authentication, the API Connect user record is updated from the backend identity provider.

   If multiple user registries are available for selection on the sign up page, then make sure that the correct registry for your API Manager account is selected. You might need to ask your administrator which user registry is appropriate for your account.

   For information on configuring user registries and making them available for API Manager login, see User registries overview and Selecting user registries for Cloud Manager and API Manager.

2. Click Sign up to complete your registration, then click Sign in to open the API Manager login page.

Results

You have created your API Connect account.

What to do next

Log in to the API Manager user interface and, depending on your role, start to work with Catalogs, consumer organizations, APIs and Products.
To access the API Manager login page in the future, use the following URL:

`https://host/manager`

where *host* is the fully qualified host name or IP address of the Management server.

Related concepts

- Working with Products

Related tasks

- Creating and configuring catalogs
- Working with consumer organizations

Related reference

- Accessing the API Manager user interface

Related information

- Developing your APIs and applications

Accessing the API Manager user interface

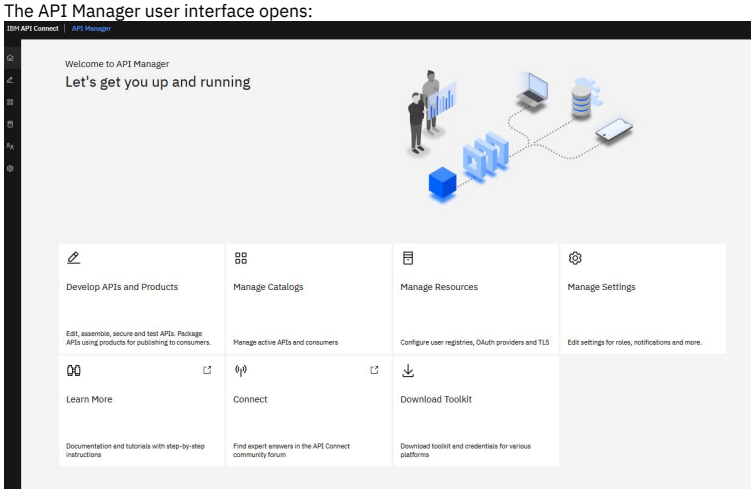
The IBM® API Connect API Manager user interface provides a range of options for working with your APIs and Products, and managing security.

To log in to the API Manager user interface, complete the following steps:

1. In a web browser, enter the following URL:

   `https://host/manager`

   where *host* is the fully qualified host name or IP address of the Management server.
2. If your cloud contains multiple user registries, then select the user registry that contains the login credentials for your API Manager account. You might need to ask your administrator which user registry is appropriate for your account.
   For information on configuring user registries and making them available for API Manager login, see User registries overview and Selecting user registries for Cloud Manager and API Manager.
3. Enter your user name and password, then click Sign in.
   The API Manager user interface opens:



Depending on your role, you can now start to work with Catalogs, consumer organizations, APIs and Products.

Searching for items in API Manager

Use the search feature in IBM® API Connect API Manager to easily locate items such as APIs, Catalogs, Spaces, applications, and subscriptions.

Search is available from any page in API Manager and can be accessed by typing in the search field in the page banner, or by clicking Search in the navigation list.

Using Search

The Search feature is available on all pages of the API Manager. To start searching, just type a string into the Search field and press Enter. Whenever you press Enter in the Search field, the Search page displays with results and additional options. If you press Enter in an empty Search field, or click Search in the navigation list, then the Search page displays a list all items. If you type a string before pressing Enter, the page displays the results from the corresponding search.

Search results include all items that contain the search string in a text field. For example, suppose you searched for "Crafted". The results might look like the following image, with items that don't match the search string exactly, but do include the string in a text field such as a title or a name.

By default, the search returns all types of items that contain the string. Sort results on a particular column by clicking the column header.

When you find an item you want to work with, click the . . . actions menu next to the item's "Modified" date to see what you can do. The list of possible actions depends on the item's type.

You can limit the search results by selecting an item type from the list that displays before the results. Click More to view additional types. If you select a type from the More menu, it replaces a type in the list.

If you want to refine the search further, click Options. On the Options panel, click Catalogs to Search and select which catalogs to search (the default is all catalogs). You can select up to 5 catalogs for a limited search.

You can also choose whether only exact matches (the complete string) are included in results, or items containing similar strings are also included.

Working with Catalogs

Products must be staged to a Catalog and then published; application developers are able to access the APIs in a Product by being members of consumer organizations to which the Product is made available. In API Connect, you can create multiple Catalogs. Catalogs are useful for separating Products and APIs for testing before you make them available to consumer organizations. The syndication feature in API Connect means that you can also publish a Product to a Space in a Catalog.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
A Catalog is a staging target, and behaves as a logical partition of the gateway and the Developer Portal. The URL for API calls and the Developer Portal are specific to a particular Catalog. In a typical configuration, an API provider organization uses a development Catalog for testing APIs under development and a production Catalog for hosting APIs that are ready for full use. A common approach is to have a development cloud with a development Catalog, a few test Catalogs and a production cloud that might have its own test Catalog.

You can use a *Space* to partition a Catalog so multiple teams can manage Products and APIs independently in a single Catalog. A Space is conceptually like a sub-catalog, except that Products and APIs in all Spaces in a given Catalog are published to the same developer portal. For more information about Spaces, see Using syndication in API Connect.

Note:
Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **`A user already exists with this email address`**.

For more information on using the Developer Portal, see Developer Portal: Socialize your APIs.

- **Creating and configuring catalogs**
  How to create and configure catalogs in IBM API Connect.
- **Configuring Product visibility at a Catalog or Space level**
  How to set a limit on the visibility for all Products that are published to a Catalog or Space.
- **Configuring sub paths for Developer Portal sites**
  You can increase the specificity of a IBM API Connect Developer Portal site URL by using sub paths.
- **Managing Catalog membership**
  Manage Catalog membership by adding new users to the Catalog and assigning roles to the users.
- **Importing a user-defined policy into a Catalog**
  You can make your user-defined policy available to API developers by importing it into an IBM API Connect Catalog, or a Space in a Catalog.
- **Importing an OpenAPI extension into a Catalog**
  You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An extension is imported into a Catalog, or a Space in a Catalog, then added to the API schema.
- **Retaining Version 5 vanity endpoint behavior in a Catalog**
  The behavior of vanity endpoints in a Catalog has changed in API Connect Version 10 compared with Version 5. However, you can modify your Catalog settings to retain the Version 5 behavior.

Creating and configuring catalogs

How to create and configure catalogs in IBM® API Connect.

Before you begin

You must have catalog create permissions to complete this task. For more information about permissions, see API Connect user roles.
Notes:

- As the user who creates the catalog, you are automatically the catalog owner and have all catalog permissions.
- If you configure a new user registry for authenticating users to the Developer Portal, you must also complete the onboarding section in the catalog, to make the registry available to Developer Portal users. See Step 14 for details.
- Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: **`A user already exists with this email address`**.

Procedure

To create and configure your catalog, complete the following steps:

1. Log in to API Manager and click Manage.
2. Create the catalog.
   You can create the catalog in either of the following ways:

Specify the owner, then configure the catalog:

a. Click Add > Create catalog.
b. In the Catalog Owner field, select the owner of the catalog; by default, you are the owner. You can specify any user who is a member of the provider organization.
c. Continue from step 3 to configure the catalog.

Send an invitation email, with an activation link, to the intended catalog owner:

a. Click Add > Invite catalog owner.
b. Enter the email address of the catalog owner, then click Invite.
c. On receipt of the email, the catalog owner must activate the catalog and configure it:
    i. Open the activation link, complete the sign up form, and sign in to the API Manager UI.
    ii. Click Manage Catalogs, click the catalog name that was provided on the sign up form, then click the Catalog settings tab.
    iii. Continue from step 6 to configure the catalog.

3. Enter the catalog Title. A Name is entered automatically and cannot be changed.
   Note: The value in the Name field is a single string that is used to identify the catalog in developer toolkit CLI commands. The Title is used for display; you can change the title but the name does not change.
   To view the CLI commands to manage catalogs, see apic catalogs.

4. Click Create.
   Your new catalog is displayed on the Manage page.
5. To continue to configure your catalog, select the catalog on the Manage page, then click the Catalog settings tab.
6. To configure the general settings for the catalog, click Overview, then proceed with the following steps:
   a. By default, the new catalog is a development catalog. To use the catalog in production, set the Production Mode slider control to the On position, then click Confirm.
      In a development catalog, all publish operations are forced, and any conflicts are resolved automatically. If you republish a previously published product version it is overwritten without warning, allowing you, for example, to repeatedly republish the same product version during testing. If spaces are enabled in a development catalog and you republish a previously published product version to a different space, it is removed from the previous space.

      However, if the catalog has been configured to require approvals, a development catalog behaves the same as any other catalog with regard to requiring approval for staging and publishing actions.

      Note: In a production catalog, the following conditions apply:
      - You cannot publish a product to the catalog if there is already a product in the catalog with the same name and version. Instead, you must create a new version of the product for publication; see Creating a new version of your product . If spaces are enabled in a production catalog, you cannot publish a product with the same name and version to more than one space in the catalog.
      - If this new product contains one or more modified APIs, you **must** create new versions of the APIs for inclusion in the product; see Creating a new version of an API definition. If the product contains a modified API and there is already a published API of the same name and version, the changes will **not** be published.

   b. To enable the partition of this catalog into spaces, set the Spaces slider control to the On position, then click Confirm.
      For more information, see Using syndication in IBM API Connect.
   c. If the catalog is the default staging catalog, move the Default slider control to the On position, then click Confirm. Then, calls to APIs that are published to the catalog can use a shorter URL that does not include the catalog name.
      Note: You can enable Default for only one catalog. If another catalog is already set as the default catalog, you will be unable to enable the setting on this catalog without first disabling the setting on the other catalog.
   d. To enable application lifecycle for applications that are created in the Developer Portal that is associated with this catalog, set the Application Lifecycle slider control to the On position, then click Confirm.
      When application lifecycle is enabled, any application that an application developer creates in the Developer Portal has Development status by default. When application testing is complete, the application developer can request to upgrade the application to production status. For more information, see Managing the application lifecycle.
7. To transfer ownership of the catalog to a different user, click Overview, then proceed with the following steps:
   a. Click Edit.
   b. From the Select owner user drop-down menu, select the user who will become the new owner.
   c. If spaces are enabled in the catalog then to also have the ownership of all spaces in the catalog transferred to the new catalog owner, select Also transfer ownership of owned Spaces.
   d. Click Save when done.

   Note:
   - To have permission to change the catalog owner, you must be either the catalog owner, or the owner of the provider organization that contains the catalog.
   - When the ownership changes, the new owner is assigned administration privileges for the catalog, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in Managing Catalog membership.
   - The ownership can be transferred only to a user who is already a member of the catalog. To transfer to a new user or another existing user, that user must previously have been added or invited to the catalog, as described in Managing Catalog membership.
8. To configure the gateway service settings for the catalog, click Gateway Services, then proceed with the following steps:
   Note: If spaces are enabled in the catalog then you configure the gateway settings for the spaces, **not** for the catalog as a whole.
   a. Click Edit.
   b. Select the gateway services that you want to use with this catalog.
      Gateway services are configured by the using the Cloud Manager UI. For details, see Registering a gateway service. If any catalog default gateway services have been configured, they will already be selected when you create a new catalog; see Configuring default gateway services for catalogs.
      The TYPE column indicates the gateway type for each gateway service listed, DataPower® Gateway (v5 compatible), DataPower API Gateway. For more information, see API Connect gateway types.

   c. Click Save to save your changes.

You publish APIs by adding them to a product and then publishing the product to a catalog. To be able to publish products to a catalog, the catalog must be assigned at least one gateway service so that the APIs in the product are available to be called at a gateway service endpoint. You can assign more than one gateway service to a catalog, and they can be of mixed types, DataPower API Gateway and DataPower Gateway (v5 compatible).

A product is gateway type specific, either DataPower API Gateway or DataPower Gateway (v5 compatible). By default, when you publish a product to a catalog, it is published to all the assigned gateway services of the same gateway type as the product, but you can choose to publish the product only to selected gateway services, of that type, at publish time; see Publishing a draft product for more information. The APIs in the product will be available to be called at all the specified gateway service endpoints.

If you select two or more Gateway services then, for analytics data to be available in the Developer Portal for this catalog, the Gateway services must all be associated with the **same** analytics service. If the set of associated analytics services for the selected gateway services includes two or more different analytics services then the Developer Portal does **not** display analytics data. An analytics service is associated with a gateway service in the Cloud Manager user interface; see Associating an analytics service with a gateway service.
Note: You can also configure the gateway service settings for the catalog by using the developer toolkit CLI; for details, see apic configured-gateway-services.

9. To specify the product lifecycle state changes for which you want to enforce approval, complete the following steps:
    a. Click Lifecycle Approvals in the catalog settings navigation pane, then click Edit.
    b. Select the required state changes, then click Save when done.
    For example, if you select Publish and leave the other check boxes cleared, approval is required when anyone attempts to publish a product, but no approval is required for any of the other lifecycle state changes.
    Note: Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
    c. To allow the originator of a product lifecycle change to approve it themselves, move the Task self approval slider control to the On position.
    Note: In a development catalog, the originator of a product lifecycle change can **always** approve it themselves. In a production catalog, the originator of a product lifecycle change can approve it themselves only if the catalog setting Task self approval is enabled.
10. Click Save.
11. To configure the publish validations that are performed when a product is published, complete the following steps:
    a. Click Publish validations in the catalog settings navigation pane.
    b. Click Edit, and select or clear the validation check boxes as required.
    The following validation options are available:
        - Publishing APIs with empty paths is not allowed.
        - Validate custom policy schema in assembly.
        - Validate the references inside the API and perform additional OpenAPI validations.
        - Publishing APIs with duplicate base paths is not allowed.
12. Click Save.
13. To configure the permissions that each role in API Manager has, complete the following steps:
    a. Click Roles in the catalog settings navigation pane.
    To see the current permissions for a role, expand the role.
    b. Click the options icon ⋮ alongside the role that you want to work with, then click Edit.
    c. Select or clear the permissions check boxes as required.
    To ensure a role can manage user-defined policies, select Manage for the Settings permission.

    To grant a role the permission to approve a specific lifecycle state change, select the state change in the Product-Approval section.

    Note: You cannot remove a permission that has been assigned to the role at the provider organization level. However, you can add permissions that haven't been assigned at the provider organization level.
    d. Click Save when done.
14. To manage the onboarding of API consumers into this catalog, complete the following steps:
    a. Click Onboarding in the catalog settings navigation pane.
    b. To select the registries that are used to authenticate users of the Developer Portal associated with this catalog, click Edit in the Catalog user registries section, select the required registries, then click Save.
    For details on how to configure a user registry, see Authenticating by using your enterprise user registry.
    Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
    c. To set a default registry, click the options icon ⋮ alongside the required registry, then select Set default.
    When an API consumer signs up to the Developer Portal, the specified registry is used by default, with the option to select another registry.

    d. To allow API consumers to complete their own sign-up process to the Developer Portal, move the Self service onboarding slider control to the On position.
    If this option is disabled, an API consumer cannot sign-up without an invitation from a consumer organization owner.
    e. To require approval for all new self service onboarding to the Developer Portal, move the Self service onboarding approval slider control to the On position.
    If this option is enabled, then any attempt by an API consumer to sign-up to the Developer Portal results in an approval request being sent. This request is displayed in the Task tab in the catalog for the associated Developer Portal, from where the request can be approved or declined.
    Note that Self service onboarding approval is honored whenever a consumer organization is created. So, even if an API consumer has already been approved to access a Developer Portal with a specific consumer organization, if they then try to create another consumer organization in the same Developer Portal, their request will still go through the approval process.

    Note: To enable self-service onboarding approval, you must also complete the following tasks:
        - Configure the user registry to require the user's email address.
        - Configure the OIDC apps to send the user's email address for approval.
    f. To edit the timeout period for the self service onboarding task, click Edit in the Self service onboarding task timeout section, specify the timeout period required, and click Save. The timeout period for this task covers both the email activation link and, if Self service onboarding approval is enabled, the approval process. The default timeout period for the self service onboarding task is 72 hours.
    Note that expired self service onboarding tasks are cleared only once per day in the Developer Portal. Therefore, any API consumers with expired onboarding tasks have to wait until these tasks are cleared before attempting to sign-up again.
    g. To configure the ability for API consumers to invite collaborators and assign them to roles, click Edit in the Consumer invitation and roles section, select the options required, and click Save.
    h. To configure the timeout for activation links that are sent in email invitations to application developers, click Edit in the Invitation Timeout section, specify the timeout length, then click Save.
    i. To override the timeouts set by consumer organization invitations, enable the Override consumer organization's invitation timeout with catalog invitation timeout setting by moving the slider to the On position.
    The catalog's own timeout setting will be used instead.

15. To specify the user registries that are used to secure access to APIs in this catalog, and to add the user registry to the Sandbox catalog, complete the following steps:
    a. Click the Catalog settings tab, and then select API User Registries.
    b. Click Edit and select the user registries you want to use.

For details on how to configure a user registry, see Authenticating by using your enterprise user registry.

c. Click Save when done.

Note:

- Only LDAP and Authentication URL registries can be used to secure access to APIs; therefore, only registries of these types are listed and available for selection.
- If spaces are enabled in the catalog then you specify the user registries for the spaces, **not** for the catalog as a whole. If you specify a user registry in one space in a catalog, it is deployed to the gateway services across all spaces in that catalog.
  For more information about enabling and managing spaces, see Using syndication in API Connect

.

16. To specify the OAuth Providers that can be used to secure access to APIs in this catalog, complete the following steps:

Note:

- If you want to specify an OAuth provider for a catalog, ensure that the Sandbox catalog is configured to use the same OAuth provider
- Any resources that the OAuth provider references, such as API user registries or TLS client profiles, must also be enabled in the catalog
- If spaces are enabled in the catalog then you specify the OAuth providers for the spaces, **not** for the catalog as a whole. If you specify an OAuth provider in one space in a catalog, it is deployed to the gateway services across all spaces in that catalog.
  For more information about enabling and managing spaces, see Using syndication in API Connect.

a. Click OAuth Providers in the catalog settings navigation pane.

b. Click Edit, then select the OAuth providers that you want to use.
To configure an OAuth provider, see either Configuring a native OAuth provider using Cloud Manager or Configuring a native OAuth provider using API Manager.

c. Click Save when done.

17. Optional: Configure vanity API endpoints for your catalog.

For any API, there are two possible endpoints to consider:

- The gateway endpoint at which the API is invoked.
- The endpoint that is visible to the consumer in the Developer Portal.

If you do not configure vanity endpoints, these two endpoints are the same, and point to the gateway endpoint at which an API is invoked.
To have an endpoint visible to the consumer that is different to the gateway endpoint, you configure a vanity endpoint. A vanity endpoint represents the endpoint by which an API is known externally; that is, the endpoint that is published to the Developer Portal and is used by an application developer to invoke the API. The way in which the gateway endpoints for an API are determined depends on the OpenAPI version of the API, as follows:

(OpenAPI 2.0): For any enforced API in API Connect, the gateway endpoint formats for the API are as follows:

- If the OpenAPI definition has no **`host`** field, the API endpoint has the following format:

  `https://`*`gateway_service_host`*`/`*`provider_organization`*`/`*`catalog`*`/`*`basepath`

- If the OpenAPI definition has a **`host`** field (for example, **`petstore.com`**), the host is appended to the path after the *provider_organization*/*catalog* segments, and the API endpoint has the following format:

  `https://`*`gateway_service_host`*`/`*`provider_organization`*`/`*`catalog`*`/`*`host_field_value`*`/`*`basepath`

  Note: This is a change in behavior from IBM API Connect Version 5.0, where the **`host`** field is not included in the API endpoint.

In these endpoint URLs, the variables are as follows:

- *gateway_service_host* is the host name of the gateway service on which the API is running.
- *provider_organization* is the value of the **`name`** field of the provider organization that contains the catalog in which the API is published.
- *catalog* is the value of the **`name`** field of the catalog in which the API is published.
- *basepath* is the value of the **`basepath`** field in the OpenAPI definition for the API.
- *host_field_value* is the value of the host field in the OpenAPI definition for the API.

For details on configuring the host server URL in an OpenAPI 2.0 API definition, see Specifying the host for an API.

API Gateway only **(OpenAPI 3.0):** If the API is enforced by the DataPower API Gateway, the value specified for the first **`url`** entry in the **`servers`** array in the OpenAPI definition of the API is used to determine the **`basepath`** element of the API endpoint, as follows:

a. If a relative URL is specified for the server URL, that value is used as-is for the **`basepath`**, and the API endpoint has the following format:

`https://`*`gateway_service_host`*`/`*`provider_organization`*`/`*`catalog`*`/`*`basepath`

b. If a full URL is specified for the server URL, the scheme (**`http://`** or **`https://`**) is ignored by the DataPower API Gateway, and the remaining host name and **`basepath`** are used to form the API endpoint as follows:

`https://`*`gateway_service_host`*`/`*`provider_organization`*`/`*`catalog`*`/`*`host_name`*`/`*`basepath`

In these endpoint URLs, the variables are as follows:

- *gateway_service_host* is the host name of the gateway service on which the API is running.
- *provider_organization* is the value of the **`name`** field of the provider organization that contains the catalog in which the API is published.
- *catalog* is the value of the **`name`** field of the catalog in which the API is published.
- *host_name* is the value of the host name in the server URL.
- *basepath* is the value of the **`basepath`** derived from the server URL.

For details on configuring the server URL in an OpenAPI 3.0 API definition, see Defining servers for an API, Defining servers for a Path, and Defining servers for an operation.
Note: You can change your catalog settings to use the IBM API Connect Version 5.0 endpoint behavior, including support for a feature known as *host to catalog mapping*, whereby an API can be invoked without having to include the provider organization or catalog in the URL path; see Retaining Version 5 vanity endpoint behavior in a Catalog.
To configure vanity API endpoints, so that you can publish endpoints that are different to these gateway endpoints, complete the following steps:

a. Click API Endpoints in the catalog settings navigation pane. The Vanity API Endpoint page opens.

b. Click Edit.

c. To display the current API base endpoint settings, select Display vanity endpoint.

d. Select the required Preference setting. Choose one the following settings:

- Catalog priority: Any host defined in the OpenAPI definition of the API is ignored and all API endpoints always point to the endpoints that you define here. For example, suppose you define a vanity API endpoint, with Catalog priority preference, to be **`https://prod.acme.com/`**. Then the API endpoint will be **`https://prod.acme.com/`*`basepath`*.
  The way in which the **`basepath`** is determined depends on the OpenAPI version of the API, as follows:

- **(OpenAPI 2.0):** The `basepath` is taken from the `basepath` field in the OpenAPI definition.
- API Gateway only **(OpenAPI 3.0):** The `basepath` is taken from the first `url` entry in the `servers` array in the OpenAPI definition, ignoring any scheme (`http://` or `https://`) and host name.

  API priority: The host defined in the OpenAPI definition for the API takes precedence. For example, suppose you define a vanity API endpoint, with API priority preference, to be `https://api.acme.com/`. Then the following rules determine the endpoint that is used when an API is called:
  - If the OpenAPI definition has a host defined, `test.acme.com` for example, then that value is used to determine the API endpoint. For example: `https://test.acme.com/basepath`.
  - If the OpenAPI definition has no host defined, the API endpoint will be `https://api.acme.com/basepath`, as determined by the vanity API endpoint setting.

  The way in which any defined host is determined depends on the OpenAPI version of the API, as follows:
  - **(OpenAPI 2.0):** The host is taken from the `host` field in the OpenAPI definition.
  - API Gateway only **(OpenAPI 3.0):** The host is taken from the host name in the first `url` entry in the `servers` array in the OpenAPI definition.

e. Supply one or more endpoints, as follows:
- If you selected Catalog priority, click Add to enter one or more endpoints URLs, and an optional summary for each. Any of the endpoint URLs can be used to invoke an API.
- If you selected API priority, enter the endpoint URL, and an optional summary.

Note: After configuring your catalog to support your vanity URL, you must configure your external network to map the vanity endpoints to the corresponding gateway endpoints. This typically includes the following:
- A DNS mapping to ensure that the vanity host resolves to the gateway.
- Additional URL routing for the API as required.

For example, if your gateway has a default domain name of `apigw.dc.zone.mycompany.com` and an IP address of `29.12.141.150`, then the generic DNS configuration might look like:

```
api.acme.com. CNAME apigw.dc.zone.mycompany.com.
```

or

```
api.acme.com. A 29.12.141.150
```

Consult your network administrator and the DNS provider's documentation to do this configuration.

18. To configure the TLS client profiles that are used with this catalog, complete the following steps:
    Note: If spaces are enabled in the catalog then you configure the TLS client profiles for the spaces, **not** for the catalog as a whole. If you configure a TLS client profile in one space in a catalog, it is deployed to the gateway services across all spaces in that catalog.
    For more information about enabling and managing spaces, see Using syndication in API Connect.

    a. Click TLS Client Profiles in the catalog settings navigation pane.
    b. Click Edit, then select the TLS client profiles that you want to use with this catalog.
    For details on how to create a TLS client profile, see Creating a TLS client profile.
    c. Click Save when done.
19. To configure the Developer Portal, complete the following steps:
    a. Click Portal in the catalog settings navigation pane, then click Create.
    b. Select the portal service that you want to use with this catalog.
    Portal services are configured by using the Cloud Manager UI. For details, see Registering a portal service.
    c. Optionally, override the default generated portal URL.
    The URL must have the following format:

    ```
    https://host_name.portal.com
    ```

    where *host_name.portal.com* must resolve to the IP address of the Developer Portal machine.
    For example:

    ```
    https://myhost.mydomain.com
    ```

    Important:
    - The URL must be unique across all catalogs.
    - Mixed-case names are supported for resource names; however, portal endpoint URLs can support only lower-case. If a provider organization is created with a mixed-case name, you must edit the default-generated portal URL to make the provider organization name lower-case in the URL.

    To use sub paths in your site URL, see Sub paths for Developer Portal sites.
    d. Select the portal service that you want to use with this catalog.
    For more information on using the Developer Portal, see Developer Portal: Socialize your APIs.

    e. Click Create to save your changes.

    Note: The account profile for the owner of the catalog must have an email address specified to receive the creation notification, otherwise the Developer Portal creation operation will fail. To check whether your account profile has an email address, and specify a value, complete the following steps:

    a. Click the User icon, and then select My Account.
    b. Check the contents of the Email field, and specify a value if necessary, then click Save.

    Note: If you want to edit the portal service or URL, or delete the portal site, return to this portal settings page, click the options icon, and click Edit or Delete.
20. To define catalog properties, complete the following steps:
    a. Click Properties in the catalog settings navigation pane, then click Create.
    b. Enter the property Name and Value, then click Create.

    The properties that you define can be referenced in any of the API definitions in this catalog. It is also possible to define properties that are specific to an API definition; see Setting API properties. For information on how to reference a property in an API definition, see Variable references in API Connect.
    Note:
    - If you define a catalog property of the same name as an API property, the API property takes precedence over the catalog property.
    - If you change the value of a catalog property, any API that references that property must be republished for it use the new value.
21. Optional: To configure the visibility of products that are published to a catalog or a space, see Configuring Product visibility at a Catalog or Space level.

What to do next

Import your own policies into the catalog. For details, see Importing a user-defined policy into a catalog.
Note: When you import a policy, its implementation is imported into all Gateway devices associated with the Gateway service that hosts the catalog. Additionally, API Connect modifies some object names and file locations to mark them with the appropriate catalog name and policy version.
For more information about how to create and manage user-defined policies, see User defined policies.

Configuring Product visibility at a Catalog or Space level

How to set a limit on the visibility for all Products that are published to a Catalog or Space.

Before you begin

To configure Product visibility at the Catalog or Space level, you must be assigned to a role that has the Settings > Manage permission for that Catalog or Space. For more information on assigning Catalog permissions to a role, see Creating and configuring catalogs.

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in API Connect.

About this task

The availability of a Product to users is determined by both the visibility setting (that covers which users can view the Product), and the subscribability setting (that covers which users can subscribe to the Product). When you create a draft Product, you typically define the visibility and subscribability in the Product document; see Specifying the visibility of your Product, and Creating a draft Product for details.
Note: After a Product is published, its visibility and subscribability can be changed in the Catalog in the API Manager UI. For more information, see Changing the availability of a Product.
You can define a Catalog or Space setting to control the visibility of all of the Products that are published to that Catalog or Space, as described in the following instructions. When this Catalog or Space setting is enabled, it can be used for the following two purposes:

- The setting can ensure that any Product published to that Catalog or Space, cannot have a visibility setting that exceeds the visibility limit set at the Catalog or Space level.
- The setting can also be used as the default visibility setting for all Products that are published to that Catalog or Space.

When you're publishing a Product, the visibility setting can come from the following four places:

1. Product override; set during the API Manager UI sequence to publish a Product. If this setting is used, it supersedes any visibility setting in the draft Product.
2. Product document; set in the draft Product.
3. Catalog or Space setting. If Product visibility is enabled in the Catalog or Space, and the visibility is not set by a Product override or in the Product document, then the Catalog or Space visibility setting is used as the default setting. This setting can also be used as a limiter, to ensure that any Product that is published cannot exceed the Catalog or Space visibility setting.
4. Default setting. If there are no other visibility settings, then the following default visibility and subscribability settings are used:

```
visibility: {
    view: {
        enabled: true,
        type: 'public',
        orgs: []
    },
    subscribe: {
        enabled: true,
        type: 'authenticated',
        orgs: []
    }
}
```

The following table shows which Product level visibility settings are allowed, if the Catalog level visibility setting is enabled.

Table 1. Relationship between the Catalog level visibility settings, and the Product level visibility settings.

Catalog level visibility setting	Product level visibility setting
Public.	Public, Authenticated, or Custom.
Authenticated.	Authenticated, or Custom.
Custom.	Custom. Note that the Product level visibility organization and group URLs are limited by the Catalog level organization and group URLs.

The following table shows how the Product subscribe setting is limited by the Product visibility level. The basic premise to remember, is that if a Product is to be available for subscription, it must be visible. For example, if you want the subscription audience to be `authenticated`, the visibility audience cannot be `custom`.

Table 2. Relationship between the Product visibility settings, and the Product subscribe settings.

Product visibility level	Product subscribe setting
Public.	Authenticated, or Custom.
Authenticated.	Authenticated, or Custom.
Custom.	Custom. Note that the Product subscribe organization and group URLs are limited by the Product visibility organization and group URLs.

Note:

- Changes to the Catalog visibility setting affects only future Product publishes. There is no change to Products that are already published.
- Any updates to the visibility of a Product are always verified against the Catalog visibility setting, as the Catalog visibility setting sets the maximum visibility level that is allowed. Therefore, if Product visibility is enabled in a Catalog, and the visibility setting of a Product is less restrictive than is set in the Catalog, then the Product publish to that Catalog will fail. For example, if the Catalog visibility setting is **`authenticated`**, then any Products that are published to that Catalog cannot have a visibility setting of **`public`**.
- You can also modify the Catalog visibility settings by using the API Connect REST APIs; see the API Connect REST API documentation.

Procedure

- To modify the Catalog or Space visibility settings by using the API Manager UI, complete the following steps.
  1. Open the Catalog that you want to work with.
  2. Click the Catalog settings tab.
  3. Click Product visibilty in the side navigation bar.
  4. Set the Enable product visibility toggle to On.
  5. Select the organization or groups that you want the Products in the Catalog to be visible to.
     You can select from the following options:
     - Public - Products are visible to all users.
     - Authenticated - Products are visible only to users who have successfully authenticated.
     - Custom - Products are visible only to the consumer organizations and groups that you specify.
  6. Click Save.
- To modify the Catalog or Space visibility settings by using the developer toolkit CLI, complete the following steps.
  1. Log in to your API Connect Management server as a member of a provider organization, by using the following command:

     ```
     apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
     ```

     Where:
     - *mgmt_endpoint_url* is the platform API endpoint URL.
     - *user_id* is the user ID that you want to log in with. The user ID must be a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
     - *password* is the password that is associated with the supplied user ID.
     - *identity_provider* is the identity provider that is used to authenticate the supplied user ID.

     For example:

     ```
     apic login --server platform-api.myserver.com --username myuser --password mypassword --realm provider/myldap
     ```

     You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

     ```
     apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
     ```

     For example,

     ```
     apic identity-providers:list --scope provider --server myserver.com --fields title,realm
     total_results: 2
     results:
       - title: API Manager User Registry
         realm: provider/default-idp-2
       - title: Corporate LDAP user registry
         realm: provider/corporate-ldap
     ```

     The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
     For full details on how to log in to your management server from the CLI, see Logging in to the management server.

  2. Define your visibility settings by creating a YAML file with the following structure:

     ```
     allowed_product_visibility: {
         view: {
             enabled: true_or_false,
             type: 'visibility_type',
             org_urls: [],
             group_urls: []
         },
         subscribe: {
             enabled: false,
             type: 'authenticated'
         }
     }
     ```

     Where, in the `view` section of the `allowed_product_visibility` property:
     - `enabled` - can be set to `true` or `false`. The default setting is `false`.
       If set to `true` then any Product that is published to that Catalog or Space can have only a visibility that is the same or more restrictive than is configured in the `type` setting. If no visibility is set in the Product document, then the Catalog or Space setting is used as the default for the publish. If you want to always use the Catalog or Space setting as the default, then you must ensure that a Product visibility is not set.
       Note: If you are creating draft Products, and no visibility is set, then the value from the `allowed_product_visibility` field in the Sandbox Catalog is used as the default.
       If set to `false`, the feature is disabled, and the Product visibility settings that are defined in the Product document are used.
     - `type` - if `enabled` is set to `true`, then this property specifies the maximum visibility of the Products that are published to this Catalog or Space. Valid settings are:
       - `public` - Products are visible to all users.
       - `authenticated` - Products are visible only to users who have successfully authenticated.
       - `custom` - Products are visible only to the consumer organizations and groups that are specified in `org_urls` and `group_urls`.
     - `org_urls` - if `type` is set to `custom`, then this property specifies the URLs of the consumer organizations that you want the Products to be visible to. Otherwise, leave empty.
     - `group_urls` - if `type` is set to `custom`, then this property specifies the URLs of the consumer organization groups that you want the Products to be visible to. Otherwise, leave empty.

     The `subscribe` section cannot be edited at the Catalog or Space level. Subscribability is defined in the Product document `subscribe` section, and must be at least as restrictive as `authenticated`. For more information, see Specifying the visibility of your Product.

  3. Modify your Catalog settings by using the following command:

     ```
     apic catalog-settings:update --server mgmt_endpoint_url --catalog catalog_name --org organization_name filename
     ```

     Where *filename* is the name of the YAML file that you created in step 2.

Note: If Spaces are enabled for the Catalog, then the visibility settings must be set at the Space level by using the `--space` option.

4. Confirm your updated Catalog settings by using the following command:

```
apic catalog-settings:get --server mgmt_endpoint_url --catalog catalog_name --org organization_name --output -
```

Related tasks

- Creating and configuring catalogs

Configuring sub paths for Developer Portal sites

You can increase the specificity of a IBM® API Connect Developer Portal site URL by using sub paths.

By adding additional text to the URL of a Developer Portal site, you can create more site permutations for an organization. Developer Portal site URLs must be unique across all Catalogs.

Important:

- Sub paths are case-sensitive.
- The Developer Portal host name must contain only **lowercase** characters.

You can create a Developer Portal in the following format:

`https://host_name.portal.com/sub_path_1/sub_path_2/.../sub_path_n`

Where *sub_path* can be any text that you want to implement to specify your site URL. There is no limit to the number of sub paths that you can have for the first site that you want to create.
However, you must not create overlaps in the naming of your URLs. A URL of one site must not be able to be confused with pages belonging to another site. For example, if you create the following site URL:

`https://host_name.portal.com/bank/safe/dollar`

Then, you cannot have the following site URL permutations:

`https://host_name.portal.com/bank/safe`

or

`https://host_name.portal.com/bank/safe/dollar/cent`

However, you can lengthen or shorten a site URL if you change a single sub path value, or the *host_name.portal.com* value. For example, if you created the following site URL:

`https://host_name.portal.com/bank/safe/dollar`

Then, you can create the following site URL permutations:

`https://host_name.portal.com/bank/euro`

and

`https://host_name.portal.com/bank/euro/vault/cent`

Important: If you already have a site URL that ends in *sub_path* texts, you cannot create a new site URL with the sub path text at the start.
For example, if you create the following site URL:

`https://host_name.portal.com/bank`

then, you cannot create the following site URL:

`https://bank.host_name.portal.com`

Managing Catalog membership

Manage Catalog membership by adding new users to the Catalog and assigning roles to the users.

Before you begin

To manage Catalog members in the API Manager, you must be assigned to a role that has the Member > Manage permission for that Catalog. For more information on assigning Catalog permissions to a role, see Creating and configuring catalogs.

About this task

The Catalog owner can invite members of the provider organization work with the Catalog in the API Manager. The invited users are called "Catalog members" and are different from consumer users who are members of consumer organizations.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.

2. Click the Members tab.
3. To add an existing user, click Add > Add member. To add a new user, click Add > Invite member.
   If you are using the Add member option, you can add any user who is already a member of another Catalog, or of a Space, in the provider organization, and is neither currently a member of this Catalog, nor the Catalog owner.
4. Enter the email address of a new user, or search for, and select, an existing user.
5. Select the roles that you want to assign to the user.
   For details of the roles and the default permissions assigned to them, see API Connect user roles. For details on how to create your own roles, see Creating custom roles.

   If Spaces are enabled in the Catalog, any role that you assign to the user at the Catalog level is assigned automatically to the user in all Spaces in that Catalog. Furthermore, if a user had originally been assigned a role only in a specific Space in a Catalog and you subsequently assign the user that role at the Catalog level, the Space specific role assignment is lost and the user now has that role in all Spaces in the catalog.
   Note: You can subsequently change the roles assigned to a user by selecting or clearing the appropriate check boxes alongside that user on the Members tab. If a user was originally added to the provider organization, rather than to the Catalog itself, the following conditions apply:
   - Any role assigned to the user at the provider organization level is assigned automatically to the user in all Catalogs, and cannot be removed at the Catalog level.
     For details on adding a user to a provider organization, see Adding provider organization users and assigning roles.
   - In the Catalog, the user has the permissions that are configured for the role at the Catalog level.
   - Any role that hasn't been assigned to the user at the provider organization level can be assigned to the user at the Catalog level.
6. If you are adding an existing user, click Add. If you are adding a new user, click Invite.

Importing a user-defined policy into a Catalog

You can make your user-defined policy available to API developers by importing it into an IBM® API Connect Catalog, or a Space in a Catalog.

About this task

For instructions, see Defining, packaging, and publishing a catalog-scoped policy for the API Gateway.

Importing an OpenAPI extension into a Catalog

You can extend the OpenAPI specification by adding either a JSON or YAML extension schema to an API. An extension is imported into a Catalog, or a Space in a Catalog, then added to the API schema.

Before you begin

You must possess Catalog edit permissions to complete this task.

About this task

This task describes how to import an OpenAPI extension into a Catalog by using the browser based API Manager user interface. Alternatively, you can use the developer toolkit CLI; see Working with OpenAPI extensions.

For an example of an extension schema file, see Referring to an extension in an API definition.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
3. Click the Catalog settings tab if you are working in a Catalog, or the Space settings tab if you are working in a Space.
4. Click Gateway services.
5. Alongside the Gateway service that you want to work with, click the options icon and then click View extensions.
6. Click Upload then upload the extension file as indicated, either by dragging and dropping the file, or by browsing and selecting it. The extension schema file will be either a JSON or YAML file.
7. Click Upload when done.

Results

The user-defined extension is now imported into a Catalog or Space, and can be used when developing an API by using the API editor in either the API Designer or the API Manager user interface.

What to do next

Add the OpenAPI extension to your APIs as required; see Adding an OpenAPI extension to an API.

Retaining Version 5 vanity endpoint behavior in a Catalog

The behavior of vanity endpoints in a Catalog has changed in API Connect Version 10 compared with Version 5. However, you can modify your Catalog settings to retain the Version 5 behavior.

About this task

A vanity endpoint represents the endpoint by which an API is known externally; that is, the endpoint that is published to the Developer Portal and is used by an application developer to invoke the API.

You can modify your Catalog settings to define vanity endpoints that retain the API Connect Version 5 behavior.

You can modify the Catalog settings in either of the following ways:

- Use the API Manager user interface.
- Use the developer toolkit CLI.

You can also use the API Connect REST APIs; see the API Connect REST API documentation.

Procedure

- To modify the Catalog settings to retain Version 5 vanity endpoint behavior by using the API Manager user interface, complete the following steps:
  1. Click API Endpoints in the Catalog settings navigation pane. The Vanity API Endpoint page opens.
  2. Click Edit, then select v5 legacy behavior.
  3. To use the API Connect URLs, select Use gateway URLs. To configure a vanity endpoint, select Display vanity endpoint then provide the following information:
     - Endpoint: The URL of your vanity endpoint.
     - Gateway Uri: The URL of the gateway service that is enabled in the Catalog and is associated with the vanity endpoint; select from the list provided.
     - Summary: An optional description of the vanity endpoint.
- To modify the Catalog settings to retain Version 5 vanity endpoint behavior by using the developer toolkit CLI, complete the following steps:
  1. Log in to your API Connect Management server as a member of a provider organization, by using the following command:

     ```
     apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
     ```

     where:
     - *mgmt_endpoint_url* is the platform API endpoint URL.
     - *user_id* is the user ID that you want to log in with. The user ID must be a member of a provider organization. This is an ID that you could also use to log in to the API Manager user interface.
     - *password* is the password associated with the supplied user ID.
     - *identity_provider* is the identity provider that is used to authenticate the supplied user ID.

     For example:

     ```
     apic login --server platform-api.myserver.com --username myuser --password mypassword --realm provider/myldap
     ```

     You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

     ```
     apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
     ```

     For example,

     ```
     apic identity-providers:list --scope provider --server myserver.com --fields title,realm
     total_results: 2
     results:
       - title: API Manager User Registry
         realm: provider/default-idp-2
       - title: Corporate LDAP user registry
         realm: provider/corporate-ldap
     ```

     The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
     For full details on how to log in to your management server from the CLI, see Logging in to the management server.

  2. Define the configuration for your vanity endpoint by creating a YAML file with the following structure:

     ```
     legacy_endpoint_behavior: enabled
     v5_endpoint_substitution_behavior:
       base_endpoints:
         - endpoint: vanity_enpoint_url
           description: description
           gateway_service_url: gateway_service_url
       unenforced_api_base_endpoint: unenforced_api_base_endpoint
     ```

     where:

     - the `legacy_endpoint_behavior: enabled` setting specifies that the Catalog retains API Connect Version 5 behavior. A Catalog can support either Version 5 behavior or Version 10 behavior but not both.
     - *vanity_enpoint_url* is the URL of your vanity endpoint.
       Note: API Connect Version 5 supports only the host name in a vanity endpoint URL; a URL of the form `https://host_name/path` is not supported. Therefore the vanity endpoint URL that you supply here must contain only a host name; for example, `https://vanity.com`.
     - *description* (optional) is a description of your vanity endpoint.
     - *gateway_service_url* is the URL of the gateway service that is enabled in the Catalog and is associated with the vanity endpoint.
       Note: You can determine the gateway service URL by using the following command:

```
apic configured-gateway-services:list --scope catalog --catalog catalog_name --server mgmt_endpoint_url --org organization_name
```

where:
- *gateway_service_name* is the name of the gateway service.
- *catalog_name* is the name of the Catalog.
- *organization_name* is the name of the provider organization that contains the Catalog.

The command output lists all the gateway services that are enabled in the Catalog, together with their gateway service URLs.

- *unenforced_api_base_endpoint* (optional) is a custom API URL that specifies the URL for APIs that are deployed to a third party gateway.

Note: `base_endpoints:` is an array. You can configure multiple vanity endpoints by defining multiple `endpoint:` entries.

3. Modify your Catalog settings by using the following command:

```
apic catalog-settings:update --server mgmt_endpoint_url --catalog catalog_name --org organization_name filename
```

where *filename* is the name of the YAML file that you created in step 2.

4. Confirm your updated Catalog settings by using the following command:

```
apic catalog-settings:get --server mgmt_endpoint_url --catalog catalog_name --org organization_name --output -
```

Managing the application lifecycle

By using the application lifecycle capability in IBM® API Connect, the applications that are subscribed to use the API initially have Development status. When application testing is complete, the application developer can request to upgrade the application to Production status.

Configuring API Connect to use the application lifecycle

To configure API Connect to use the application lifecycle capability, you must enable Application Lifecycle in the Catalog Settings. See step 6.d in Creating and configuring catalogs.

Using application lifecycle in the Developer Portal

If application lifecycle is enabled for the Catalog associated with your Developer Portal, then when you register a new application it initially has Development status.

When you have completed your application testing, you can request to upgrade your application to Production status; For details, see Upgrading a Development application to Production status.

Approving an application upgrade request

A request to upgrade a Development application to Production status is approved or declined in the API Manager UI; for details, see Approving application upgrade requests.

Related information

- Registering a gateway service
- Applications in the Developer Portal
- Exploring APIs and Products in the Developer Portal

Approving application upgrade requests

To approve or decline requests by application developers to upgrade a Development application to Production status, use the Approvals tab in the Catalog that is associated with the IBM® API Connect Developer Portal from which the application upgrade request was submitted.

Before you begin

To approve application upgrade requests in a Catalog, you must either be the owner of the associated API provider organization, or you must be assigned a user role that has Subscription and Application Approvals > Manage permission in the Catalog. For information on configuring permissions for a Catalog, see Creating and configuring catalogs. For information on assigning user roles, see Managing Catalog membership.

About this task

If application lifecycle is enabled for a Catalog, any application that an application developer registers in the associated Developer Portal initially has Development status. When application testing is complete, the application developer can request to upgrade the application to Production status.

Procedure

To approve application upgrade requests, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. Click the Tasks tab, then locate the application upgrade request that you want to deal with.

An application upgrade request has an Application Lifecycle label.
3. Click Approve or Decline as required. If you decline the request, a Reason for declining window is displayed, giving you the option to supply the reason why you declined the request. Click Decline and send when done.
A notification email is sent to the originator of the request.

Using syndication in API Connect

With the API Connect syndication feature, you can partition your catalogs into *Spaces*. Each space is used by a different API provider development team and has its own set of management capabilities relating specifically to the APIs that the associated team publishes to that space, enabling each team to manage their APIs independently.

When you stage or publish an API to a catalog that has spaces enabled, you specify the space within that catalog that you want to stage or publish to. However, application developers that access the Developer Portal for the catalog are unaware of the space partitioning of the catalog and see the APIs as a coordinated offering.

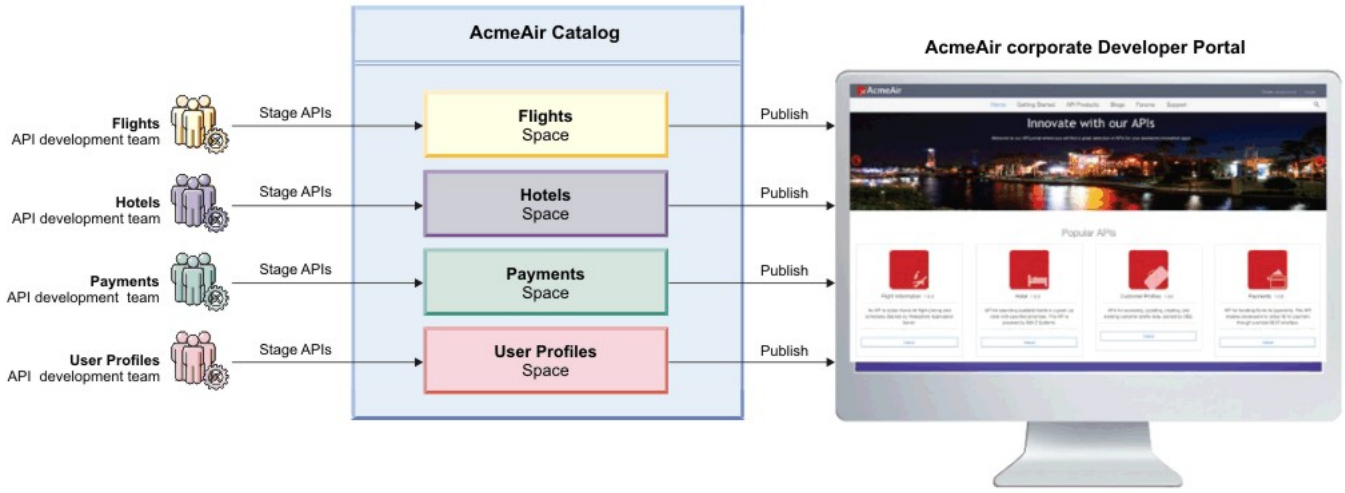
Each space has its own product lifecycle management, subscription approvals, and analytics data. You use space specific access control to restrict user access to each space; for example, a developer in the Flights team is able to stage APIs only to the Flights space.

Example

A travel company, AcmeAir, has separate API provider development teams for each of the following areas of their business:

- Flights
- Hotels
- Payments
- User Profiles

They have an AcmeAir corporate catalog, with its associated corporate Developer Portal. They partition their catalog into separate spaces, one for each development team.



Spaces are described in detail in the following sections:

- **Enabling Spaces in a Catalog**
To use the syndication feature in IBM API Connect, you must enable Spaces in any Catalog in which you require syndication capabilities.
- **Creating, modifying, and deleting Spaces**
You use the API Manager user interface to create a new Space in a Catalog, modify the configuration, and owner of a Space, and delete a Space from a Catalog.
- **Working with Spaces**
If Spaces are enabled in an IBM API Connect Catalog, you can select a specific Space to work with in the API Manager user interface. This enables you to manage the Products and APIs that are specific to that Space, and control user access to the Space.

Enabling Spaces in a Catalog

To use the syndication feature in IBM® API Connect, you must enable Spaces in any Catalog in which you require syndication capabilities.

Before you begin

If Spaces are enabled for a Catalog, Products (and their associated APIs) can be published only to a Space within that Catalog. For information about publishing, see Staging a Product for publishing by using the API Designer, and see Publishing APIs for publishing by using the developer toolkit.

Note: You cannot enable Spaces in a Catalog to which one or more Products have already been staged or published. If the Catalog does contain staged or published Products, first complete the following steps for each Product in the Catalog:

1. Remove all staged Products.
2. Retire, then remove, all published Products.

After enabling Spaces, re-stage or re-publish Products, and re-create application subscriptions, as required.

About this task

By default, Spaces are disabled in a Catalog. You enable Spaces by modifying the Catalog settings. After Spaces are enabled for a Catalog, a default Space is automatically created that inherits all of the configuration settings from the Catalog.

Note: You can also enable Spaces by using the developer toolkit CLI; for details, see apic spaces.

Procedure

To enable Spaces in a Catalog, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Catalog settings tab.
3. Set the Spaces slider control to the On position.
4. In the Enable Spaces window, click Enable.

Results

Spaces are enabled for your Catalog, and a default Space is created that automatically inherits all of the configuration settings from the Catalog. The Catalog owner is also the owner of the default Space.

What to do next

You can modify Space settings, and create more Spaces; see Creating, modifying, and deleting Spaces and Working with Spaces for more information.

Creating, modifying, and deleting Spaces

You use the API Manager user interface to create a new Space in a Catalog, modify the configuration, and owner of a Space, and delete a Space from a Catalog.

Before you begin

Navigate to the Spaces page of the Catalog containing the Space you want to work with, by completing the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.

Note: You can also enable Spaces by using the developer toolkit CLI; for details, see apic spaces.

Procedure

- To create a new Space, complete the following steps:
  1. Click Add > Add a new Space.
  2. In the Space Details window, enter the Title of the Space and, optionally, a descriptive summary. A Name is entered automatically.
     Note: The value in the Name field is a single string that is used to identify the Space in developer toolkit CLI commands. The Title is used for display.
     To view the CLI commands to manage Spaces, see apic spaces.
  3. Click Create to create the Space.

  Note:
  - As the user who creates the Space, you are automatically the Space owner and have all Space permissions.
  - When you create a new Space, it does not automatically inherit all of the configuration settings from the Catalog. So you should check and modify the settings as necessary.
- To modify the details of a Space, complete the following steps:
  1. On the Spaces page, select the Space whose details you want to modify, then click Space settings.
  2. Use the options in the navigation pane to modify the details as required.
     The following types of resource can be enabled in a Space:
     - Gateway services
     - API user registries
     - OAuth providers
     - TLS client profiles

     Note:
     - If you enable a resource of any of the following types in one Space in a Catalog, it is deployed to the gateway services across all Spaces in that Catalog:
       - API user registries
       - OAuth providers
       - TLS client profiles
     - If you deploy any of the following types of resource to a gateway in a Space, it is actually deployed to the gateway at Catalog scope:
       - Custom policy
       - Global policy
       - Global policy pre-hook
       - Global policy post-hook
       - Gateway extension
- To change the owner of a Space, complete the following steps:
  1. On the Spaces page, select the Space whose details you want to modify.
  2. Click the Space settings tab to open the Space details page, then click Edit.
  3. From the Select owner user drop-down menu, select the user who will become the new owner.

4. Click Save when done.

Note:

- To have permission to change the Space owner, you must be either the Catalog owner, the Space owner, or the owner of the provider organization that contains the Space.
- When the ownership changes, the new owner is assigned administration privileges for the Space, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in Managing Space membership.
- The ownership can be transferred only to a user who is already a member of the Space or Catalog. To transfer to a new user or another existing user, that user must previously have been added or invited to the Space, as described in Managing Space membership, or to the Catalog, as described in Managing Catalog membership.

- To delete a Space, complete the following steps:

1. On the Spaces page, click the Manage icon for the Space that you want to delete.
2. Click Delete, then click Confirm to confirm deletion.

Note:

- You cannot delete a Space to which one or more Products have been staged or published.
- If there is only one Space in a Catalog, you cannot delete it, but you can disable Spaces for the Catalog.
  However, you cannot disable Spaces in a Catalog to which one or more Products have already been staged or published. If any of the Spaces in the Catalog do contain staged or published Products, first complete the following steps for the Products in each of the Spaces:
  1. Remove all staged Products.
  2. Retire, then remove, all published Products.

  Then, to disable Spaces for the Catalog, complete the following steps:
  1. Click the Catalog settings tab in the Catalog.
  2. Set the Spaces slider control to the Off position.

  After disabling Spaces, re-stage or re-publish Products, and re-create application subscriptions, as required.

Working with Spaces

If Spaces are enabled in an IBM® API Connect Catalog, you can select a specific Space to work with in the API Manager user interface. This enables you to manage the Products and APIs that are specific to that Space, and control user access to the Space.

Procedure

To work with a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.

What to do next

For details of the Space management tasks, see the following subtopics:

- **Managing Products in a Space**
  If Spaces are enabled in an IBM API Connect Catalog, you can use the API Manager user interface to separately manage the Products that are staged or published to each Space.
- **Managing subscription requests in a Space**
  If Spaces are enabled in an IBM API Connect Catalog, you can approve, or deny, individual requests to subscribe to Plans in Products that are published to the Space.
- **Managing application subscriptions in a Space**
  If Spaces are enabled in an IBM API Connect Catalog, you can use the API Manager user interface to separately manage the application subscriptions to Plans in the Products that are published to a Space.
- **Managing Space membership**
  If Spaces are enabled in an IBM API Connect Catalog, you can manage the members within the Space. You manage Space membership by adding new users to the Space and assigning roles to the users.
- **Managing user access in a Space**
  If Spaces are enabled in an IBM API Connect Catalog, you can manage the access that users have within the Space. You manage access by specifying the permissions that are assigned to user roles.
- **Managing Gateways in a Space**
  If Spaces are enabled in a Catalog, you can separately control which Gateway services are used by each of the Spaces in the IBM API Connect Catalog.

Managing Products in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can use the API Manager user interface to separately manage the Products that are staged or published to each Space.

Before you begin

To manage Products in a Space in the API Manager UI, a user must be assigned a role that has the Products > Manage permission. For more information on assigning Space permissions to a role, see Managing user access in a Space.

Procedure

To manage the Products that are staged or published to a specific Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.

What to do next

For details of the management tasks that you can perform on the Products in the Space, see Working with Products.

Managing subscription requests in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can approve, or deny, individual requests to subscribe to Plans in Products that are published to the Space.

Before you begin

To manage Space subscription requests in the API Manager UI, a user must be assigned a role that has the Subscription Approvals > Manage permission. For more information on assigning Space permissions to a role, see Managing user access in a Space.

Procedure

To manage the subscription requests in a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.
4. Click the Tasks tab.
   The Tasks page lists only the subscription requests for the selected Space.

What to do next

For details on how to manage approvals, see Approving Product lifecycle and subscription requests.

Managing application subscriptions in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can use the API Manager user interface to separately manage the application subscriptions to Plans in the Products that are published to a Space.

Before you begin

To manage application subscriptions in the API Manager UI, a user must be assigned a role that has the Subscriptions > Manage permission. For more information on assigning Space permissions to a role, see Managing user access in a Space.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.
4. In the navigation pane of the API Manager UI, click the Applications tab.
   All applications that have been registered in the Developer Portal associated with this Space are listed.
5. Click the options icon ⋮ alongside the application you want to work with, then click View Subscriptions.

What to do next

For details of the management tasks that you can perform on application subscriptions in a Space, see Working with application subscriptions.

Managing Space membership

If Spaces are enabled in an IBM® API Connect Catalog, you can manage the members within the Space. You manage Space membership by adding new users to the Space and assigning roles to the users.

Before you begin

To manage Space members in the API Manager UI, a user must be assigned a role that has the Child > Manage permission for the Catalog that contains the Space. For more information on assigning Space permissions to a role, see Managing user access in a Space.

About this task

Note: You can add the same user to two or more Spaces and assign different roles in each, allowing a user to have differing levels of access in different Spaces.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.
4. Click the Members tab.
5. To add an existing user, click Add > Add member. To add a new user, click Add > Invite member.
   If you are using the Add member option, you can add any user who is already a member of another Space, or of a Catalog, in the provider organization, and is neither currently a member of this Space, nor the Space owner.
6. Enter the email address of a new user, or search for, and select, an existing user.
7. Select the roles that you want to assign to the user.
   For details of the roles and the default permissions assigned to them, see API Connect user roles. For details on how to create your own roles, see Creating custom roles.

   If a user was originally added either to the provider organization, or to the Catalog that contains the Space, rather than to the Space itself, the following conditions apply:
   - Any role assigned to the user at the provider organization or Catalog level is assigned automatically to the user in all Spaces, and cannot be removed at the Space level.
     For details on adding a user to a provider organization, see Adding provider organization users and assigning roles.

     For details on adding a user to a Catalog, see Managing Catalog membership.
   - In the Space, the user has the permissions that are configured for the role at the Space level.
     For details on configuring role permissions for a Space, see Managing user access in a Space.
   - Any role that hasn't been assigned to the user at the provider organization or Catalog level can be assigned at the Space level

   Note: You can subsequently change the roles assigned to a user by selecting or clearing the appropriate check boxes alongside that user on the Members page.
8. If you are adding an existing user, click Add. If you are adding a new user, click Invite.

Managing user access in a Space

If Spaces are enabled in an IBM® API Connect Catalog, you can manage the access that users have within the Space. You manage access by specifying the permissions that are assigned to user roles.

Before you begin

To manage Space permissions in the API Manager UI, a user must be assigned a role that has the Child > Manage permission for the Catalog that contains the Space. For information on adding users to a Space and assigning user roles, see Managing Space membership.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.
4. Click the Space settings tab.
5. Click Roles, click the options icon ⋮ alongside the role whose permissions you want to modify, then click Edit.
6. Select the required permissions, then click Save.

Related tasks

- Creating custom roles

Related information

- API Connect user roles

Managing Gateways in a Space

If Spaces are enabled in a Catalog, you can separately control which Gateway services are used by each of the Spaces in the IBM® API Connect Catalog.

Before you begin

To manage Space Gateway settings in the API Manager UI, a user must be assigned a role that has the Child > Manage permission for the Catalog that contains the Space. For information on adding users to a Space and assigning user roles, see Managing Space membership.

About this task

Note: If Spaces are enabled in a Catalog then you configure the Gateway settings for each Space separately, **not** for the Catalog as a whole.

Procedure

To manage the Gateway settings for a Space, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Click the Spaces tab.
3. Select the space that you want to work with.
4. Click the Space settings tab.
5. Click Gateway services.
   The Gateway services that are currently enabled for the Space are listed.
6. To change the Gateway service configuration for the Space, click Edit, then select which Gateway services you want the Space to use.
7. Click Save to save your changes.

Related information

- Configuring the initial Gateway service

Working with Products

In IBM API Connect, Plans and APIs are grouped together in Products.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
Information about managing Products in the API Manager UI can be found in the following topics.

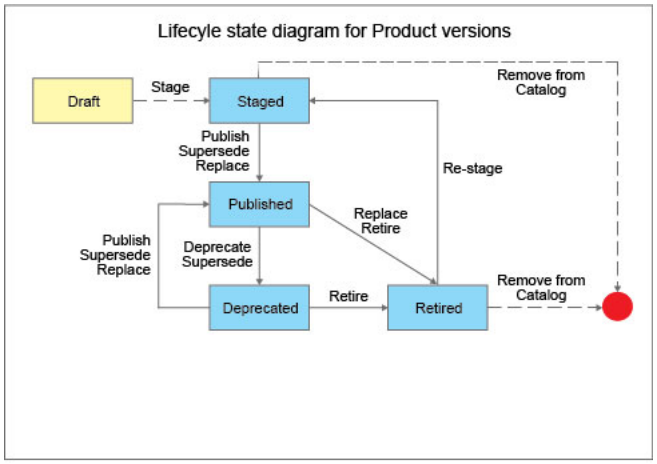
- **The Product lifecycle**
  When you manage your Product versions, you move them through a series of lifecycle states. From initially staging a Product version to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. The syndication feature in IBM API Connect means that Product lifecycle states can also be managed within Spaces in the associated Catalog.
- **Monetizing your Products**
  In addition to offering free Plans for your customers to use your Products, you can also define Plans that automatically bill your customers who are using your Products in IBM API Connect.
- **Managing your Products**
  You can manage your products in API Manager by using the Products page of the associated catalog. In this view, you can move the products through their lifecycle, display analytics information, and control who can see or subscribe to the products. The syndication feature in IBM API Connect means that products can also be managed by using the Products tab of the associated space.

The Product lifecycle

When you manage your Product versions, you move them through a series of lifecycle states. From initially staging a Product version to a Catalog, through to publishing to make the Product version available to your application developers, and to eventual retiring and archiving. The syndication feature in IBM® API Connect means that Product lifecycle states can also be managed within Spaces in the associated Catalog.

Product lifecycle state diagram

The following diagram shows the possible lifecycle states for a Product version, and the Product management operations that move a Product version from one lifecycle state to another. For example, the Retire operation moves a Product version from the Published to the Retired state.



Note: The same Product lifecycle states apply irrespective of whether your Product is managed within a Catalog, or within a Space in a Catalog. For more information about the syndication feature, see Using syndication in API Connect.
If approval is required for a Product management operation, an approval request is sent and the Product version moves to the pending state. When the request is approved, the operation is completed and the Product version moves to the next lifecycle state. If approval is not required, the operation is completed immediately.
Note: Approval is not required for the following lifecycle state transitions:

- Retired to Staged.
- Deprecated to Published.

For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.
Note:

- Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon ⋮ alongside the Product that you want to work with, and selecting View Approval History.

The following sections describe the various lifecycle states for a Product version.
Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

Draft

The draft state for a Product or API is when a Product or API definition is not deployed and is not associated with any Catalog.

Staged

When you stage a Product, a copy of the Product version is deployed to the target Catalog. Staged is the initial state when you publish a Product. When a Product is in the staged state, it is not yet visible to, or subscribable by, any developers. For more information about staging a Product, see Staging a Product.

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests. For information on publishing a Product, see Publishing a Product.

Published

When you publish a Product, a fixed copy of the Product version is deployed to the target Catalog. The Product version is visible to, and subscribable by, the targeted developers or communities. When a Product is published in a Catalog, the visibility and subscription settings can be changed for the published version of that Product. Any further changes require a new version of the Product to be staged and published before they take effect.

If you replace a Published Product with a Staged or Deprecated Product, the replacement Product is published, and the replaced Product is retired.

If you supersede a Published Product with a Staged or Deprecated Product, the superseding Product is published, and the superseded Product is deprecated.

For more information about publishing a Product, see Publishing a Product.

Note: If you want to update a Product, you must republish it. The `products:update` command only updates a Product's metadata.

Deprecated

When you deprecate a Product, the Product version is visible only to developers whose applications are currently subscribed. No new subscriptions to the Plans in the Product are possible. For more information about deprecating Products, see Deprecating a Product.

A Product is also deprecated if you supersede it with another Product. For more information, see Superseding a Product with another Product.

Retired

When you retire a Product, the Product version can neither be viewed nor can its Plans be subscribed to, and all of the associated APIs are taken offline. For more information about retiring Products, see Retiring a Product.

A Product is also retired if you replace it with another Product. For more information, see Replacing a Product with another Product.

Related concepts

- Working with Catalogs

Related tasks

- Managing your Products
- Working with Spaces

Monetizing your Products

In addition to offering free Plans for your customers to use your Products, you can also define Plans that automatically bill your customers who are using your Products in IBM® API Connect.

API Connect includes a subscription billing feature that allows API providers to define pricing Plans in their API Products, and monetize their API offerings. If a Product contains a pricing Plan, API consumers must enter their payment information into the Developer Portal before they can subscribe to that Plan.

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

You can define both free Plans and paid Plans within the same Product. To charge your customers more for greater access to your APIs, you can create multiple Plans with different tiers of prices and rate-limits. For example, an introductory Plan might have a low monthly price and restrictive rate-limiting, while a higher tier Plan might couple a higher monthly price with less restrictive rate limiting controls. Your customers can then select the Plan that is appropriate for their usage rates. As your customers have different needs for the use of your Products, so you can offer different levels of service. For example, a smaller customer might need a rate of only 5 API calls per hour, whereas a larger company might need a rate of 1000 API calls per hour. You can create the Plans that are appropriate for your customers, and they can subscribe to the service and pricing Plan that is appropriate for them.

Restriction: The following limitations apply to the subscription billing feature:

- Billing and monetization requirements vary by country, where IBM and the independent cloud billing provider, Stripe, cannot guarantee every country requirement is supported. Country requirements change, and are out of scope for this document. Some limitations on storing and passing fields to the Stripe billing provider might exist, when fields are unique to a subset of countries. For example, the passing of a billing address from API Connect to Stripe, which is required by some EU countries due to GDPR and taxation laws, is not currently supported.
- Two data center disaster recovery configurations including the optional billing microservice are not supported at this time.
- After a configured billing resource is in use by a Catalog, it might not be possible to remove it. Do not add your Stripe test keys to production Catalogs. If you want to prototype with your non-production Stripe keys, create a separate temporary Catalog.
- If an API consumer deletes their payment method, or their credit card expires, or their card is out of credit after they have initially subscribed and prepaid for their first month, there is no mechanism to alert the provider organization or suspend API access automatically in response. Provider organizations can monitor their Stripe dashboards for non-payment events and, if necessary, notify the consumer, suspend the application for that subscription, or cancel the API Connect subscription manually. The URL of the API Connect subscription is saved in the metadata of the Stripe subscription object, to assist in reverse lookups.
- Suspending an application does not automatically pause the billing cycles of the subscriptions for that application.
- You can republish a Product that includes billing only when the Catalog Production mode is set to On.

Though much of the procedure is the same for setting up a paid Plan and a free Plan, there are some extra actions that are needed before you can start defining paid Plans. See the following topics for more information:

1. First, you must configure a billing integration resource for your provider organization; see Adding a billing integration resource.
2. Before you can publish a Product Plan with pricing to a Catalog, you must add your billing integration resource to that Catalog; see Adding a billing integration resource to a Catalog.
3. Also before you can publish to a Catalog, an administrator must enable the Stripe payment method module in the Developer Portal for that Catalog; see Configuring Stripe in the Developer Portal.
4. After the Catalog and Developer Portal are configured, you can create Products that include pricing Plans; see Defining a Product with billing integration.

Note:

- Before you can create billing resources in provider organizations, the billing microservice must be enabled on your management system by your system administrator. See Configuring monetization on VMware and Configuring monetization on Kubernetes for more information.
- You can also create and manage billing integration resources by using the developer toolkit CLI. For more information, see Command-line tool reference for the developer toolkit.

Related tasks

- Publishing a Product
- Managing your Products

Related reference

- Troubleshooting your billing configuration

Adding a billing integration resource

To enable the monetization of your Product Plans, you must add a billing integration resource in your IBM® API Connect provider organization that defines the configuration data needed to synchronize with an external subscription billing system.

Before you begin

To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See Firewall requirements on Kubernetes and Firewall requirements on VMware for more information about this requirement.

Before you can create billing resources in provider organizations, the billing microservice must be enabled on your management system by your system administrator. See Configuring monetization on VMware and Configuring monetization on Kubernetes for more information.

You must either be the provider organization owner, or have `Settings: Manage` permissions, to complete this task.

You must have an account with Stripe to be able to complete this task. If you do not already have a Stripe account, you can set one up at: www.stripe.com.

About this task

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system. When API consumers subscribe to Product Plans with billing in the Developer Portal, they set up automatic payments through a credit card registered with Stripe. To enable this billing process, you must first specify your Stripe account information by creating a billing integration resource. The subscription payment amount is processed from the account that is provided by the API consumer, and credited to the account that you provide as the API provider.

Important: Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.

Procedure

To create a billing integration resource, complete the following steps:

1. In the API Manager, click Resources.
2. Ensure that you are in the provider organization to which you want to add the billing integration.
3. Select Billing, and then click Add.
   The Add billing integration page for Stripe integration is displayed.
4. Enter a Title for your billing integration.
5. The Name is auto-generated based on the Title that you enter, and is a single string that can be used in developer toolkit CLI commands.
6. Enter the Publishable key and Secret key for your Stripe account.
   Refer to your Stripe dashboard to get your publishable key and secret key; see https://dashboard.stripe.com/apikeys.
7. Click Add to create your billing integration resource.

Results

Your new billing integration resource is listed in the table on the Billing dashboard. The table also displays the state of the job queue for the billing integration. Any issues with the job queue are displayed in this table. See Billing integration resource job queues for more information.

What to do next

To be able to publish monetized Product Plans, you must add the billing integration resource to your Catalog, and to the Sandbox Catalog. See Adding a billing integration resource to a Catalog for details.

Related tasks

- Defining a Product with billing integration

Related reference

- Troubleshooting your billing configuration

Billing integration resource job queues

Billing integration resources use job queues to manage the bulk Product lifecycle operations in IBM® API Connect.

Subscription billing systems such as Stripe, typically provide APIs that support updating a single resource at a time. However, IBM API Connect has Product lifecycle operations that can modify many resources in a single transaction, such as migrating a set of subscriptions from one Product to another Product. In order to bridge the mismatch between these two APIs, API Connect adds a job queue to each billing integration resource in your provider organization. Jobs on this queue are responsible for

mapping bulk operations to a series of API calls against the subscription billing system. Jobs run one at a time, in the order that they were created. When a job completes it's deleted from the queue, and the next job runs. When all of the jobs in the queue complete running and the queue is empty, the data model of the subscription billing system should be up-to-date with the changes made in API Connect.

There are potential issues when moving this type of synchronization to background jobs. Network connectivity, outages at the billing provider, or rolling of the API keys of the billing system, could all cause the job queue to be blocked if the root issue does not resolve by itself in the span of an hour. If a job encounters an error, such as a network connectivity error, the job is retried up to five times over the course of 70 minutes, before changing the state of the job from Running to either Blocked or Failed. Only when the state of the head job is blocked or failed, will the Retry Blocked Jobs action appear in the options menu. So if an error is shown but the job is still attempting to run, you need to wait up to 70 minutes before the Retry Blocked Jobs action appears.

The job queue status can be viewed in the table on the Billing dashboard. The table lists any blocked jobs, and the error that is blocking them. If the error indicates that your API keys are no longer valid, you must update your billing resource to contain the correct values. After the root cause is resolved, you can click the ⋮ options menu on the row of the billing integration resource, and click Retry Blocked Jobs. This action retries the jobs on the job queue. After a few minutes the number of jobs should go down. Note that you can click the Refresh icon ↻ to refresh the results from the server.

Related concepts

- Monetizing your Products

Adding a billing integration resource to a Catalog

To be able to publish monetized Product Plans, you must add a billing integration resource to your Catalogs.

Before you begin

You must have created a billing integration resource. See Adding a billing integration resource for details.

You must either be the provider organization owner, or have `Settings: Manage` permissions, to complete this task.

About this task

Before you can publish any Products that include pricing Plans, you must add the billing integration resource to the Catalog where you will create the monetized Products. When API consumers subscribe to Product Plans with billing in the Developer Portal, they set up automatic payments through a credit card registered with Stripe. To enable this billing process, after you have specified your Stripe account in a billing integration resource, you must then apply that resource to your Catalogs. The subscription payment amount is processed from the account that is provided by the API consumer, and credited to the account that is provided in the billing integration resource.

Note:

- Before you can publish a monetized Product Plan to a Catalog, the Stripe payment method module in the Developer Portal must also be enabled; see Configuring Stripe in the Developer Portal.
- Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.

Procedure

To add a billing integration resource to a Catalog, complete the following steps:

1. In the API Manager, click Manage.
2. Select the Catalog tile on the Manage page that you want to update, then click the Catalog settings tab.
3. Select Billing, and then click Edit.
4. Select the billing integration resource that you want to apply to the Catalog, and click Save. You should apply only one billing integration resource to each production Catalog.
   The billing integration resource is now listed in the Billing section.
   Note: More than one billing resource can be applied to the Sandbox Catalog.

Results

The billing integration resource is added to the Catalog.

What to do next

The Stripe payment method module must be enabled in the Developer Portal for that Catalog; see Configuring Stripe in the Developer Portal for details.

After both the Catalog and the Developer Portal are configured, you can create Products with pricing Plans. See Defining a Product with billing integration for more information.

Related concepts

- Considerations when changing a Product lifecycle with billing integration

Related tasks

- Defining a Product with billing integration

Related reference

- Troubleshooting your billing configuration

Defining a Product with billing integration

To monetize your APIs, you must create Product Plans with pricing information that allow API consumers to subscribe to your Plans with integrated billing.

Before you begin

You must have a billing integration resource, and this resource must be added to the Catalog that you want to publish your Product to, as well as to the Sandbox Catalog. See Adding a billing integration resource, and Adding a billing integration resource to a Catalog for details.

Also, the Stripe payment method module must be enabled in the Developer Portal for the Catalog; see Configuring Stripe in the Developer Portal for details.

About this task

When you decide to charge customers to use your API Products, your customers have different requirements for the number of API calls that they are willing to pay for. You can solve this issue by defining Plans at different levels of service, and at different costs.

Within your Product, you can define free Plans as well as Plans that contain billing integration. The following instructions guide you through defining a Plan with pricing information. For more detailed information about creating Products, see Creating a draft Product and Editing a draft Product.

Important: Products that contain a billing integration cannot be re-staged into non-production Catalogs. To experiment with billing on the Sandbox Catalog, or any other non-production Catalogs, you must publish new versions of your monetized Product. This practice aligns with how you have to work with production Catalogs currently, as they also don't support the re-staging of existing versions of the same Product.

Procedure

To add pricing information to a Product Plan, complete the following steps:

1. In the API Manager, click Develop.
2. Select the Products tab.
3. Select the Product that you want to add billing integration to, or create a new draft Product by clicking Add > Product and following the wizard.
4. With the Product open and the Design tab selected, select Plans in the navigation menu.
5. In the Billing integration section, select the billing integration resource that you want to apply to the Product.
   In the Plans section, you can then either edit the Default Plan to add pricing information, or create a new Plan with pricing information.
6. If this is a new Plan, enter a Title, and if you require approval for someone to subscribe to this Plan also select the Approval checkbox.
7. In the Plan pricing section, complete the following steps to add the pricing information:
   a. Change the toggle to On for Plan pricing.
      The Plan pricing definition section is displayed.
   b. Optional: If you want to include any free trial days in your Plan, select Include free trial days, and then enter the number of trial days that a subscriber can use the Plan without charge, after which their billing cycle begins.
   c. Select the Currency for the billing process use.
   d. Enter the Price per month to bill the subscriber for.
      If the selected currency supports fractional units, enter the price including any fractional units, such as cents.
8. Optional: In the Plan rate limits section, set the rate limits by entering the number of calls that are allowed per unit of time.
   Remember that this rate limit applies only to this Plan. You can set multiple rate limits, at second, minute, hour, day, and week time intervals. You can override this Plan rate limit for specific endpoints if you prefer, adding a different rate limit or enabling unlimited calls to it. You can also create more Plans with different rate limits. For more information about rate limits, see Understanding rate limits for APIs and Plans.
9. Optional: Select the Hard limit checkbox if you want the gateway to reject any calls beyond the specified rate, rather than adding them to a queue to be resolved when the limits reset.
10. Optional: Add more burst limits, GraphQL limits, and assembly limits, as appropriate for your APIs.
11. Complete the Plan APIs section if you want to customize the APIs and operations that are available in this Plan.
12. If you want to add more Plans to the Product, repeat the process from Step 6.
13. Click Save to save the Product.

Results

Your Product with billing integration and a pricing Plan is listed in the Products dashboard of the Develop area. If your Product doesn't appear, click the Refresh icon to refresh the results from the server.

What to do next

Stage your Product to a Catalog. For more information, see Staging a draft Product.

Related concepts

- Working with Products
- Considerations when changing a Product lifecycle with billing integration

Related reference

- Troubleshooting your billing configuration

Managing your Products

You can manage your products in API Manager by using the Products page of the associated catalog. In this view, you can move the products through their lifecycle, display analytics information, and control who can see or subscribe to the products. The syndication feature in IBM® API Connect means that products can also be managed by using the Products tab of the associated space.

Before you begin

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

For more information about the product lifecycle, see The Product lifecycle.
For more information about the syndication feature, see Using syndication in API Connect.

For information on viewing analytics that can provide insight into product and API usage and performance, see API Analytics.

For information about product billing, see Monetizing your Products.

Procedure

To manage your product, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Expand a product to see details of the APIs and plans in that product.

   Manage options are also available, depending on the state of the product. To view the manage options, click the options icon and select the options that you require.
4. To manage the lifecycle of a version of a product, click the options icon alongside the product version, and select the required lifecycle action.
5. To set the migration target of a version of a product, click the options icon alongside the product version, and select Set migration target. In the Set migration target window, select the product that you want to set as the migration target, and select Next. Then map the migration source plans to the migration target plans, and click OK.
   Restriction: If the plan is part of a product that is contained within a space in the catalog, and the migration action is being made at the space level, the plan that you are migrating subscriptions from must be located in the same space as the plan you are migrating to. If the migration action is being made at the catalog level, subscriptions can be migrated across spaces. For more information about the Space feature, see Using syndication in API Connect.

- **Considerations when changing a Product lifecycle with billing integration**
  There are special considerations for lifecycle operations for Products that contain a billing integration in IBM API Connect.
- **Publishing a Product**
  APIs become accessible when a Product is published and made visible on the IBM API Connect Developer Portal for use by application developers. A Product can be published to selected communities of application developer organizations, and the Plans within the Product can be used to tailor access and visibility further.
- **Migrating application subscriptions to another Product**
  By using the Products view within a Catalog in API Manager, you can migrate application subscriptions between Products. The syndication feature in IBM API Connect means that you can also use the Products view within a Space in a Catalog to migrate application subscriptions.
- **Migrating application subscribers to new Product versions**
  When new versions of Products are created in IBM API Connect, there are a number of ways that subscribers can be moved to the Plans of the new Product.
- **Changing the availability of a Product**
  You can change the availability of a Product and the associated Plans by using the Products view within a Catalog or Space in API Manager. The availability of a Product covers both the visibility of that Product to users, and which users can subscribe to it.
- **Deprecating a Product**
  You can deprecate a published Product by using the Manage Products page within a Catalog in API Manager. When you deprecate a Product, application developers that are already subscribed to the Product can continue to use it, but no new developers can subscribe to the Product. The syndication feature in IBM API Connect means that you can also use the Manage Products page within a Space in a Catalog to deprecate a published Product.
- **Retiring a Product**
  You can retire a published or deprecated Product by using the Manage Products page within a Catalog in API Manager. When a Product is retired, all associated

APIs are taken offline, and any subscriptions are deleted. The syndication feature in IBM API Connect means that you can also use the Manage Products page within a Space in a Catalog to retire a Product.

- **Re-staging a Product**
  If a Product in a Catalog has been retired, you can re-stage the Product so that it can re-enter the Product lifecycle. A re-staged Product moves to the Staged state, from where it can be published. The syndication feature in IBM API Connect means that you can also re-stage a Product within a Space in a Catalog.
- **Working with the subscriptions in a Product**
  You can work with the subscriptions to the Plans in a specific Product. Plan subscriptions are created by application developers in the Developer Portal.
- **Updating the gateway services for a Product**
  When you stage or publish a draft Product to a Catalog, you specify the gateway services at which the APIs in the Product are to be made available. However, you can later update the selected gateway services after a Product has been stage or published.
- **Removing a Product from a Catalog**
  You can remove a Product from a specific Catalog by using the Products view within a Catalog in API Manager. The syndication feature in IBM API Connect means that you can also use the Products view within a Space to remove a Product from a Catalog.
- **Approving Product lifecycle and subscription requests**
  To approve or decline requests to change the lifecycle state of a Product, and requests by application developers to subscribe to a Plan, use the Tasks page in the Catalog that contains the associated Product in API Manager. The syndication feature in IBM API Connect means that you can also use the Tasks page within a Space in a Catalog to approve or decline requests.

Considerations when changing a Product lifecycle with billing integration

There are special considerations for lifecycle operations for Products that contain a billing integration in IBM® API Connect.

For example, if you release new versions of your Product that contain pricing changes, you will need to define the upgrade pathways for your existing subscribers. The following scenarios detail the special considerations for Product lifecycle changes:

Deprecating
: Preparing an existing Product to be removed from production. You can define a replacement Product by using the Set Migration Target option in the Catalog. API consumers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported. For more information about deprecating Products, see Deprecating a Product.

Migrating
: Moving subscriptions from one Product to a different Product, and from a free Plan to a paid Plan, is not supported. Instead, see the superseding scenario.

Replacing
: Replacing a Product with a newer version of the same Product, and automatically upgrading all existing subscribers to the new Product. You must map which Plans from your new Product correspond to which Plans in your old Product. You cannot map a free Plan to a paid Plan in the replace operation. For more information about replacing Products, see Replacing a Product with another Product.

Retiring
: Removing a Product or Plan from production, and canceling all of the existing subscriptions and their billing cycles. For more information about retiring Products, see Retiring a Product.

Superseding
: Deprecating the old Product, and at the same time defining its migration target to a newer version of itself. Retaining the existing Product and Plan for subscribers who would like to keep it, but offering a different Product and Plan for the updated version of a Product. API consumers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported. For more information about superseding Products, see Superseding a Product with another Product.

Note: You can use the Set Migration Target option in the Catalog on a Product that isn't being deprecated. For example, if you wanted to give customers the option to move to a different Product without affecting the original Product.
While these follow the same basic procedures that are described in Managing your Products for Products and Plans without billing, there are some considerations that only apply to Products that contain paid Plans.

You cannot migrate a customer directly from a free Plan to a Plan with billing

Customers who subscribe to Plans with billing must set up their billing information in the Developer Portal. If they subscribe only to free Plans, they don't have billing accounts.

There are two ways to migrate customers from a free Plan to a paid Plan:

- Supersede the free Product Plan and provide a migration target to a paid Product Plan. This deprecates the free Product Plan so no new users can subscribe to it, and gives the customer a Migrate this subscription message in the Developer Portal to move to the new Product Plan.
- Set a migration target to a paid Product Plan. In this case the free Product Plan is not deprecated, but it still gives the customer the Migrate this subscription message in the Developer Portal for moving to the new Product Plan. With this method, new customers can still subscribe to the free Product Plan.

Both of these methods require action from the customer before they move from the free Product Plan to the paid Product Plan.
Note: Free trial days do not apply to customers when they migrate. They are available only for new customers.

You must map the existing Plans to the new Plans when you supersede or replace a Product

When you replace or supersede a Plan with billing with another Plan, you must map the Plans from the Product that is being removed to the Plans that are in the new Product. It is convenient if there are the same number of Plans for each, but you can map more than one Plan from the existing Product to one Plan in the new Product.

Billing charges are prorated when subscriptions are migrated

When customers change their subscriptions to a Plan that has a different price, their next monthly invoice will contain prorated charges, depending on how long they spent on different billing Plans. For more information about how Stripe handles proration, see https://stripe.com/docs/billing/subscriptions/prorations.

Related reference

- Troubleshooting your billing configuration

Publishing a Product

APIs become accessible when a Product is published and made visible on the IBM® API Connect Developer Portal for use by application developers. A Product can be published to selected communities of application developer organizations, and the Plans within the Product can be used to tailor access and visibility further.

Before you begin

You must stage a Product before it can be published. For more information about staging Products, see Staging a Product.
Note: If you want to publish a LoopBack project, you must publish both the APIs (by publishing the Products that contain the APIs) and the associated applications so the project can be run. For more information about publishing LoopBack applications, see Publishing APIs and applications.
To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

A community is a collection of consumer organizations, used to control which organizations have access to Products and Plans without having to assign access on an individual basis. A Product can be published to selected communities, which means that only application developers within those organizations contained within the community can see the Product on the Developer Portal and obtain application keys to access it. A Product can alternatively be published to all communities. Communities are used to restrict the visibility and accessibility of APIs, for example to particular business partners, internal organizations, or other groups of application developers.

Note: If you want to update a Product, you must republish it. The `products:update` command only updates a Product's metadata.

Procedure

You can publish a product in any of the following ways:

- Publish a new Product.
- Replace a Product with another Product.
- Supersede a Product with another Product
  For details of the ways in which you can publish a Product, see the following subtopics:

- **Publishing a new Product**
  Publish a Product from within its containing Catalog in API Manager. The syndication feature in IBM API Connect means that you can also publish a Product from within its containing Space in a Catalog.
- **Replacing a Product with another Product**
  You can replace a published Product in IBM API Connect with another Product, and automatically migrate subscribers to the new Product, by using the API Manager.
- **Superseding a Product with another Product**
  You can supersede a published Product with another Product by using the API Manager.

Publishing a new Product

Publish a Product from within its containing Catalog in API Manager. The syndication feature in IBM® API Connect means that you can also publish a Product from within its containing Space in a Catalog.

Before you begin

The Product that you are publishing must be in the Staged or Deprecated state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

You can complete this task either by using the API Designer UI application, or by using the browser-based API Manager UI.

If you publish a Product to a non-development Catalog, the Product that is published is an independent and fixed copy of the version of the Product that you chose to stage. Editing the Product through the Products page will not affect the published Product. For this reason, it is recommended that when you stage a Product, you then create a new version of the Product to edit in future, so as to avoid confusion regarding the properties of the published Product. For more information on creating new versions of your Product, see Creating a new version of your Product.

An exception is that if you publish a Product to a development Catalog, editing it through the Products page will enable you to re-stage and publish the same version of the Product. For information on how to create a development Catalog, see Creating and configuring Catalogs.

Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see Working with Catalogs.

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in API Connect.

Procedure

To publish a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon and then click Publish.
   The Confirm Visibility Settings page opens.
4. Specify the following options:
   - The users that the Product is visible to
     You can choose Public users, Authenticated users, or Custom.
     - Custom - You can use the Type to add... field to search for organizations or communities that you want your Product to be visible to.
   - Who can subscribe to the Product
     You can add or remove one or more consumer organizations or communities.
     - Custom - You can use the Type to add... field to search for organizations or communities that you want to allow to subscribe to your Product.
5. Click Publish, then click Confirm to proceed with the publish operation.
   If approval is required to publish Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is published when the request is approved. If approval is not required, the Product version is published immediately, and moves to the Published state. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your Product is in the Published state.

Your Product is published to your Catalog and available to your specified organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product.
Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

Related tasks

- Deprecating a Product

Related information

- Staging a Product

Replacing a Product with another Product

You can replace a published Product in IBM® API Connect with another Product, and automatically migrate subscribers to the new Product, by using the API Manager.

Before you begin

The Product to be replaced must be in the Published state, and the replacement Product must be in the Staged, Published, or Deprecated state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

When you replace a Product with another Product, the following actions are taken:

- The replacement Product is published.
- If the visibility and subscribability settings in the replacement Product are such that access is the same as, or less restrictive than, the original Product, the settings in the replacement Product are used. If the settings in the replacement Product are more restrictive, meaning that fewer consumer organizations can see or subscribe to the Product, the replace operation fails. For more information on visibility and subscribability settings, see Changing the availability of a Product.
- The subscribers to the original Product are migrated to the replacement Product.
  Note: Customers cannot be migrated automatically from a free Plan to a paid Plan. To move your customers from a free Plan to a paid Plan, you can supersede the product with a new product and set a migration target to the paid Plan. The customers then select a button to migrate and must enter their credit card information before the process is complete. For more information, see Considerations when changing a Product lifecycle with billing integration.
- The original Product is moved to the Retired state. Products in the Retired state are removed from the Developer Portal; they are no longer visible to the application developers, and any subscriptions to them are canceled. However, the Product can be staged again later if required.

Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see Working with Catalogs.

Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

Procedure

To replace a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to replace, click the options icon, then click Replace.
   The Replace window opens.
4. Select the replacement Product, then click Next.
5. From the drop-down list, select which Plans from your new Product correspond to Plans in your old Product.
6. Click Replace.
   If approval is required to replace Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is replaced when the request is approved. If approval is not required, the Product is replaced immediately. For information on configuring product lifecycle approvals for a catalog, see Creating and configuring catalogs. For information on approving requests, see Approving product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your new Product is in the Published state, and the original Product is in the Retired state.

Your new Product is published to your preferred organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product.
Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

Related concepts

- The Product lifecycle

Related information

- Staging a Product

Superseding a Product with another Product

You can supersede a published Product with another Product by using the API Manager.

Before you begin

The Product to be superseded must be in the Published state. The superseding Product must be in the Staged, Published, or Deprecated state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM® API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

When you supersede a Product with another Product, the following actions are taken:

- The superseding Product is published.
- If the visibility and subscribability settings in the superseding Product are such that access is the same as, or less restrictive than, the original Product, the settings in the superseding Product are used. If the settings in the superseding Product are such that access is more restrictive, meaning that fewer consumer organizations can see or subscribe to the Product, the supersede operation fails. For more information on visibility and subscribability settings, see Changing the availability of a Product.
- The original Product is moved to the Deprecated state.
- The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product. In the Developer Portal the application developers will see a Migrate this subscription message, which they can click to upgrade their subscription to the migration target.
- The deprecated product can be published again if required.

Although you can publish to a development Catalog, the development Catalog should be used only for testing purposes. Similarly, a Developer Portal created from a development Catalog must be used for testing purposes only, and not for production use. For more information on Catalogs, see Working with Catalogs.

Note: All references in this topic to a Catalog, can also be applied to a Space in a Catalog, unless specified otherwise.

Procedure

To supersede a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to supersede, click the options icon and then click Supersede.
4. Select the superseding Product and then click Next.
5. From the drop-down list, select which Plans from your new Product correspond to Plans in your old Product.
6. Click Supersede.
   If approval is required to supersede Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is superseded when the request is approved. If approval is not required, the Product is superseded immediately. For information on configuring product lifecycle approvals for a catalog, see Creating and configuring catalogs. For information on approving requests, see Approving product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your superseding Product is in the Published state.

The Product that was superseded is in the Deprecated state.

Your superseding Product is published to your preferred organizations or communities. Application developers within the groups you selected can see and use the APIs within the Product. The original Product is deprecated.
Any application developer requests to use your Product are displayed on the Approvals tab in the containing Catalog, where you can decline or accept the request.

Existing customers remain subscribed to the deprecated product, and any billing cycles for subscriptions to paid plans continue uninterrupted. For more information, see Considerations when changing a Product lifecycle with billing integration.

Related concepts

- The Product lifecycle

Related tasks

- Deprecating a Product

- Publishing a new Product

Migrating application subscriptions to another Product

By using the Products view within a Catalog in API Manager, you can migrate application subscriptions between Products. The syndication feature in IBM® API Connect means that you can also use the Products view within a Space in a Catalog to migrate application subscriptions.

Before you begin

Your Product that you are migrating subscriptions from must be in one of the following states:

- Published
- Deprecated

To complete the Product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for Products in the Catalog that contains the Product. If you have View permission for Products in this Catalog, you will have read-only access to the Product management page. For information on configuring Product management permissions for a Catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

Application developers initially subscribe their applications to one or more Plans in Products by using the Developer Portal. For more information, see Exploring APIs and Products in the Developer Portal. However, by using API Manager, for a chosen Product you can migrate one or more of its application subscriptions from one Plan to a Plan in another Product.

Procedure

To migrate Product subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon and then click Migrate Subscriptions.
4. Select the Plan from which you want to migrate subscriptions, then click Next.
5. Select the subscriptions that you want to migrate, then click Next.
6. Select the Product that you want to migrate the subscriptions to, then click Next.
7. Select the Plan that you want to migrate the application subscriptions to, and then click Migrate.

Results

The target Product has the migrated subscriptions, and those subscriptions are removed from the Product from which they were migrated.

Related tasks

- Working with Spaces

Migrating application subscribers to new Product versions

When new versions of Products are created in IBM® API Connect, there are a number of ways that subscribers can be moved to the Plans of the new Product.

About this task

You can move users to the new Plans in different ways, depending on your Product strategy, by using the options icon alongside the Product in the associated Catalog in the API Manager UI.

Procedure

To automatically migrate all existing subscribers to a new Product, for example, when you are applying fixes to a Product:

- You should Replace the original version of the Product with a new version of the Product.
  - The replacement Product is Published.
  - The original Product is Retired.
  - The subscribers to the original Product are automatically migrated to the replacement Product.

Note: Customers cannot be migrated automatically from a free Plan to a paid Plan. To move your customers from a free Plan to a paid Plan, you can supersede the product with a new product and set a migration target to the paid Plan. The customers then select a button to migrate and must enter their credit card information before the process is complete. For more information, see Considerations when changing a Product lifecycle with billing integration.

For more information about replacing Products, see Replacing a Product with another Product.

To encourage subscribers to move to a new Product, and stop new users from subscribing to the original Product, for example, if an enhancement or new feature is added:

- You should Supersede the original version of the Product with a new version of the Product.
  - The superseding Product is Published.
  - The original Product is Deprecated.
  - The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product. In the Developer Portal the subscribers will see a Migrate this subscription message, which they can click to upgrade their subscription to the migration target.
  - If the migration target is a paid Plan, subscribers must enter a payment method before they can upgrade. Upgrades by subscribers from a free Plan to a paid Plan are supported.

  For more information about superseding Products, see Superseding a Product with another Product.

To prepare a Product to be removed from production, but leave the existing subscriptions as they are:

- You should Deprecate the original Product.
  - The Product is Deprecated.
  - The application developers that are already subscribed to the now deprecated Product can continue to use it, but no new developers can subscribe to the Product.
  - You can define a replacement Product by using the Set Migration Target option in the Catalog. Application developers will then see a Migrate this subscription message in the Developer Portal that they can click to upgrade their subscription to the migration target. If the upgrade target is a paid Plan, they must enter a payment method before upgrading. Upgrades by API consumers from a free Plan to a paid Plan are supported.

  For more information about deprecating Products, see Deprecating a Product.

To give subscribers the option to move to a different Product, but without affecting the original Product:

- You can use the Set Migration Target option in the Catalog on a Product that isn't being deprecated or superseded. For more information, see Managing your Products.

Changing the availability of a Product

You can change the availability of a Product and the associated Plans by using the Products view within a Catalog or Space in API Manager. The availability of a Product covers both the visibility of that Product to users, and which users can subscribe to it.

Before you begin

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM® API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

Note: All references in this topic to a Catalog can also be applied to a Space in a Catalog, unless specified otherwise. For more information about Spaces, see Using syndication in API Connect.

Procedure

To change the availability of a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon, then click Edit visibility.
   The Visibility page opens.
4. In the Visibility section, specify the users that you want the Product to be visible to. You can choose Public to make the Product visible to all users, Authenticated to make the Product available to users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that you want the Product to be visible to.
   If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that you want the Product to be visible to. For information about how to create and manage consumer organizations and consumer organization groups, see Working with consumer organizations.

   Use the Temporarily disable visibility check box to toggle whether the Product is or is not visible, without having to change other visibility settings.
5. In the Subscribability section, specify the users that can subscribe to the Plans in the Product. You can choose Authenticated to make the Plans in the Product subscribable by users who have successfully authenticated, or Custom to specify the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product.
   If you select Custom, use the Type to add organizations field to search for the consumer organizations and consumer organization groups that can subscribe to the Plans in the Product. If you selected custom visibility, only the consumer organizations and consumer organization groups selected there are available for adding to

the custom subscribability list. For information about how to create and manage consumer organizations and consumer organization groups, see Working with consumer organizations.

Use the Temporarily disable subscribability check box to toggle whether the Product can or cannot be subscribed to, without having to change other subscription settings.

6. Click Save when done.
   The Product version is configured with the modified availability settings.
   Note:
   - Changing who can view or subscribe to a Product does not affect the settings of the draft Product. The change applies only in the Catalog that was selected in Step 1.
   - Changing who can view or subscribe to a Product does not affect existing subscriptions.
   - Any updates to the visibility of a Product are always verified against the Catalog visibility setting, as the Catalog visibility setting sets the maximum visibility level that is allowed. Therefore, if Product visibility is enabled in a Catalog, and the visibility setting of a Product is less restrictive than is set in the Catalog, then the Product publish to that Catalog will fail. For example, if the Catalog visibility setting is `authenticated`, then any Products that are published to that Catalog cannot have a visibility setting of `public`.

Results

Your Product remains in the same lifecycle state, but now with new availability settings.

What to do next

You can also set Product visibility at the Catalog or Space level. For more information, see Configuring Product visibility at a Catalog or Space level.

Related tasks

- Working with Spaces
- Configuring Product visibility at a Catalog or Space level

Deprecating a Product

You can deprecate a published Product by using the Manage Products page within a Catalog in API Manager. When you deprecate a Product, application developers that are already subscribed to the Product can continue to use it, but no new developers can subscribe to the Product. The syndication feature in IBM® API Connect means that you can also use the Manage Products page within a Space in a Catalog to deprecate a published Product.

Before you begin

The Product that you are deprecating must be in the Published state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

Procedure

To deprecate a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon, then click Deprecate.
   The Deprecate Product window opens.
4. Click Confirm to deprecate the Product.
   If approval is required to deprecate Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is deprecated when the request is approved. If approval is not required, the Product is deprecated immediately. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your Product is in the Deprecated state.

Related concepts

- Using syndication in API Connect

Related tasks

- Working with Spaces

Retiring a Product

You can retire a published or deprecated Product by using the Manage Products page within a Catalog in API Manager. When a Product is retired, all associated APIs are taken offline, and any subscriptions are deleted. The syndication feature in IBM® API Connect means that you can also use the Manage Products page within a Space in a Catalog to retire a Product.

Before you begin

The Product that you are retiring must be in the Published or Deprecated state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

Procedure

To retire a Product, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon and then click Retire. The Retire Product window opens.
4. Click Confirm.
   If approval is required to retire Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is retired when the request is approved. If approval is not required, the Product is retired immediately. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your Product is in the Retired state. It is also removed from the Developer Portal, is no longer visible to the application developers, and any subscriptions to it are deleted. You can stage the Product again later if required; for details, see Staging a Product.
If the Product has any monetized Plans, the billing cycles for any subscriptions are canceled as well. For more information, see Considerations when changing a Product lifecycle with billing integration.

Related concepts

- Using syndication in API Connect

Related tasks

- Working with Spaces

Re-staging a Product

If a Product in a Catalog has been retired, you can re-stage the Product so that it can re-enter the Product lifecycle. A re-staged Product moves to the Staged state, from where it can be published. The syndication feature in IBM® API Connect means that you can also re-stage a Product within a Space in a Catalog.

Before you begin

The Product that you are re-staging must be in the Retired state.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

You stage a Product so that the appropriate approvals, internally within the organization, can be given for it to then be published. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring Catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests. For information on publishing a Product, see Publishing a Product.

Procedure

To re-stage a Product, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon and then click Re-stage. The Re-stage Product window opens.
4. Click Confirm.
   If approval is required to stage Products in this Catalog, an approval request is sent, and the Product moves to the Pending state; the Product is re-staged when the request is approved. If approval is not required, the Product is re-staged immediately. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs. For information on approving requests, see Approving Product lifecycle and subscription requests.
   Note:
   - Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
   - Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
   - You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

Results

Your Product is in the Staged state, from where it can be published; see Publishing a new Product.

Related concepts

- Using syndication in API Connect

Related tasks

- Working with Spaces

Working with the subscriptions in a Product

You can work with the subscriptions to the Plans in a specific Product. Plan subscriptions are created by application developers in the Developer Portal.

Procedure

To work with the Plan subscriptions in a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. Alongside the Product version that you want to work with, click the options icon and then click View Subscriptions.
   A window opens listing all subscriptions to the Plans in the selected Product.
4. Alongside the subscription that you want to work with, click the options icon .
   You can then select from the following options:

- View Consumer Organization: a window opens listing the consumer organization that the subscribing application belongs to. If required, you can click the options icon ⋮ to work with the consumer organization For details of the available options, see Working with consumer organizations.
- View Application: a window opens listing the subscribing application. If required, you can click the options icon ⋮ to work with the application. For details of the available options, see Working with developer applications.
- View Plan: a window opens listing the Plan that the application is subscribed to. If required, you can click the options icon ⋮ to work with the Plan. For details of the available options, see Working with Plans.
- View Product: a window opens listing the Product that contains the Plan that the application is subscribed to. If required, you can click the options icon ⋮ to work with the Product. For details of the available options, see Managing your Products.

5. To display the Client ID's for a particular Product subscription, click the information icon ⓘ in the Application column for the subscription that you require.

Updating the gateway services for a Product

When you stage or publish a draft Product to a Catalog, you specify the gateway services at which the APIs in the Product are to be made available. However, you can later update the selected gateway services after a Product has been stage or published.

Before you begin

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM® API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

About this task

When you stage or publish a draft Product to a Catalog, you can either choose to make the APIs in the Product available at all the gateway services that match the gateway service type of the Product, either DataPower® API Gateway or DataPower Gateway (v5 compatible), or can you select specific gateway services of that type. The APIs in the Product can then be called at those gateway service endpoints.

However, after a Product has been staged or published you can later update the gateway services to change the selections.

For more information on staging and publishing a Product, see Staging a draft Product and Publishing a draft Product.

For more information on the gateway service types, see API Connect gateway types.

Procedure

To update the gateway services for a Product, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon ⋮, then click Update gateway services.
   The Select Gateway Services window opens.
4. Select the required gateway services, then click Save.

Removing a Product from a Catalog

You can remove a Product from a specific Catalog by using the Products view within a Catalog in API Manager. The syndication feature in IBM® API Connect means that you can also use the Products view within a Space to remove a Product from a Catalog.

Before you begin

The Product that you are removing from the Catalog must be in the Staged or Retired state. For more information, see Staging a Product or Retiring a Product.

To complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the catalog that contains the product. If you have View permission for products, you have read-only access to the product management page. For information on configuring product management permissions for a catalog, see Creating and configuring catalogs.

The syndication feature in IBM API Connect means that products can be contained within a space in a catalog. In this case, to complete the product management tasks that are described in this topic, you must either be the owner of the API provider organization, or be assigned Manage permission for products in the space that contains the product. For information on configuring product management permissions for a space, see Managing user access in a space.

Procedure

To remove a Product from a Catalog, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon, then click Delete.
   The Delete Product window opens.
4. Click Confirm to delete the Product.

Results

Your Product is removed from the Catalog. You can stage the Product again if required; for details, see Staging a Product.
If your Product was contained within a Space in a Catalog, your Product is removed from both the Space and the Catalog.

Related concepts

- Using syndication in API Connect

Related tasks

- Working with Spaces

Approving Product lifecycle and subscription requests

To approve or decline requests to change the lifecycle state of a Product, and requests by application developers to subscribe to a Plan, use the Tasks page in the Catalog that contains the associated Product in API Manager. The syndication feature in IBM® API Connect means that you can also use the Tasks page within a Space in a Catalog to approve or decline requests.

Before you begin

To see lifecycle change requests for a Product, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or manage Products in the Catalog that contains the Product. For information on configuring Product management permissions for a Catalog, see Creating and configuring catalogs.

The syndication feature in API Connect means that Products can be contained within a Space in a Catalog. In this case, to see lifecycle change requests for a Product, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or manage Products in the Space that contains the Product. For information on configuring Product management permissions for a Space, see Managing user access in a Space.

About this task

If approvals for Product lifecycle changes are enabled for a Catalog, then an attempt to change the lifecycle state of a Product results in an approval request being sent. This request is displayed in the Tasks page in the Catalog, from where the request can be approved or declined. The authority to approve Product lifecycle state changes is restricted to users in specified roles. For information on configuring Product lifecycle approvals for a Catalog, see Creating and configuring catalogs.
Note:

- Approval for product lifecycle state changes in a catalog is disabled by default. You must explicitly enable the product lifecycle state changes that you want to enforce.
- Product lifecycle approvals can be configured only at the Catalog level. This feature is not available at the Space level.
- You can view the history of Product lifecycle requests and approvals, by clicking the options icon alongside the Product that you want to work with, and selecting View Approval History.

If a Product is set to require approval for subscription by application developers, then an attempt by an application developer to subscribe to the Product results in an approval request being sent. This request is displayed in the Approvals tab in the Catalog that contains the associated Product in API Manager, from where the request can be approved or declined.

Procedure

To work with a Product lifecycle change request, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Click the Tasks tab, then locate the Product lifecycle change request that you want to deal with.
4. Click the Approve or Decline as required.

Note: You can approve or decline lifecycle change requests only if you have a user role that has permission to approve Product publish requests in the Catalog in which the changes will take place.
For information on configuring Product management permissions for a Catalog, see Creating and configuring catalogs.

To work with a subscription request, complete the following steps.

5. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
6. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
7. Click the Tasks tab, then locate the subscription requests that you want to deal with.
   Note: To see subscription approvals, you must either be the owner of the API provider organization, or you must be assigned a user role that has permission to view or approve subscription requests. For information on creating and assigning user roles, see Administering user access.
8. Click the Approve or Decline as required.
   Note: The options to approve or decline a subscription request are available only if you are assigned a user role that has permission to approve subscription requests. For information on configuring permissions for a Catalog, see Creating and configuring catalogs For information on assigning user roles, see Managing Catalog membership.

Results

For a subscription request, an email is sent confirming whether you approved or declined the request. For a request to change the lifecycle state of a Product, no email is sent.

Related concepts

- Using syndication in API Connect

Related tasks

- Working with Spaces

Working with APIs

For a specific Product, you can work with the APIs that are contained in the Plans in that Product.

Procedure

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
To work with the APIs in a Product, complete the following steps:

1. List the Plans in the Product:
   a. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   b. If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
      i. Click the Spaces tab.
      ii. Select the space that you want to work with.
   c. If the Products page is not already displayed, click the Products tab.
   d. Alongside the Product version that you want to work with, click the options icon, then click View Plans.
      A window opens listing the Plans in the Product.
2. Alongside the required Plan, click View APIs.
   A window opens listing the APIs in the Plan.
3. Alongside the required API, click the options icon.
   You can then select from the following options:
   - Take Offline: if you take an API offline, the API is no longer available to be called. When required, you can make the API available again by selecting Online.
   - View Products: a window opens listing all the Products that contain the API. If required, you can click the options icon alongside a Product to work with that Product. For details of the available options, see Managing your Products.
   - View Plans: a window opens listing all Plans, across all Products, that contain the API. If required, you can click the options icon alongside a Plan to work with that Plan. For details of the available options, see Working with Plans.

Working with Plans

After a Product has been published to a Catalog, you can work with the Plans in that Product.

Before you begin

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
For details on configuring a Plan in a Product, see Editing a draft Product.

For details on Publishing a Product to a Catalog, see Publishing a Product.

To complete the tasks that are described in this topic, you must either be the owner of the API provider organization, or you must be assigned a role that has either the Product > Manage or the Product > View permission for the Catalog. For more information, see Adding provider organization users and assigning roles and Managing Catalog membership. For information on configuring Product management permissions for a Catalog, see Creating and configuring Catalogs.

Procedure

To work with Plans in a Catalog, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. If the product that you want to work with is contained within a space, select the required space by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   The Products page of the space opens, and all of the products available in that space are displayed.
3. Alongside the Product version that you want to work with, click the options icon and then click View Plans.
   A window opens listing all the Plans in the selected Product.
4. Alongside the Plan that you want to work with, click the options icon .
   You can then select from the following options:
   - View Product: a window opens listing the Product that contains the Plan. If required, you can click the options icon to work with the Product. For details of the available options, see Managing your Products.
   - View APIs: a window opens listing the APIs in the Plan. If required, you can click the options icon alongside an API to work with that API. For details of the available options, see Working with APIs.
   - View Subscriptions: a window opens listing the subscriptions to the Plan. If required, you can click the options icon alongside a subscription to work with that subscriptions. For details of the available options, see Working with application subscriptions.

Working with consumer organizations

Manage the consumer organizations that access your APIs and Plans when their users sign up to use the IBM® API Connect Developer Portal.

About this task

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
Application developers in consumer organizations access the Developer Portal and sign up to use the Plans in the Products you create in the API Manager. You can create more than one consumer organization and each organization has one owner. Application developers are added through the Developer Portal UI.

You can also create groups of consumer organizations. A consumer organization group is an efficient way of grouping application developers together to publish Products to, making it easier to control who has access to the Plans in the Products.

Use the following tasks to administer your consumer organizations and consumer organization groups:

- **Creating a consumer organization**
  If you have permission to manage developers, you can create new consumer organizations in a Catalog. consumer organizations contain the application developers that subscribe to use the APIs that are published to the Catalog
- **Editing a consumer organization**
  You can change the title of an IBM API Connect consumer organization, and also transfer ownership to a new organization owner.
- **Deleting a consumer organization**
  When you delete a consumer organization, all of the resources in that organization are removed from the cloud. However, the account of the organization owner remains in IBM API Connect.
- **Working with the applications in a consumer organization**
  You can work with the developer applications that have been registered in a consumer organization.
- **Working with the subscriptions in a consumer organization**
  You can work with the Plan subscriptions that have been created by the developer applications registered in a consumer organization.
- **Working with consumer organization groups**
  A *consumer organization group* is a collection of consumer organizations, and provides an efficient way of controlling who can see, and subscribe to, the APIs in your Products. By using a consumer organization group, you can define this access for all the developers in the organizations in that group in a single operation, rather than having to define access for the organizations separately.
- **Sending messages to consumer organization owners**
  You can send email messages to the owners of consumer organizations. You can also style the email to match your business theme.

Creating a consumer organization

If you have permission to manage developers, you can create new consumer organizations in a Catalog. consumer organizations contain the application developers that subscribe to use the APIs that are published to the Catalog

Before you begin

Your IBM® API Connect Developer Portal must be enabled to perform this task. For more information, see Creating and configuring Catalogs.
To complete this task, you must be defined as a member in a catalog or space (if spaces are enabled), and you must be assigned a role that has the Consumer-Org > Manage permission. For more information, see Adding provider organization users and assigning roles and Managing catalog membership.

About this task

You can create a consumer organization in either of the following ways:

- Invite a user to be a consumer organization owner, in which case they receive an invitation email with an activation link, and can complete the creation of the consumer organization and specify the title.
- Create the consumer organization yourself, specifying the owner and title. There are two ways you can specify the owner:
  - Specify the user ID of an existing user.
  - If you are using a Local User Registry, specify a new user ID; the user is added to the registry and becomes the owner of the consumer organization.

If Spaces are enabled in your Catalog, when you create a consumer organization, it is added to the Catalog and its Spaces. For more information about enabling Spaces, see Using syndication in API Connect.

Procedure

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. To invite a user to be a consumer organization owner, complete the following steps:
   a. Click Add > Invite organization owner.
   b. Enter the email address of the organization owner.
   c. Click Invite.
   The new consumer organization is added to the list, and an email invitation is sent to the owner; by using the activation link, the owner specifies the title of the organization. The status is shown as `Pending` until the recipient of the email clicks the link in the email to complete the creation of their Developer Portal account, after which the status changes to `Enabled`.
5. To create the consumer organization yourself, complete the following steps:
   a. Click Add > Create organization.
   b. Enter the organization Title. A Name is entered automatically.
   The value in the Name field is a single string that is used to identify the consumer organization in developer toolkit CLI commands. The title is used for display.
   To view the CLI commands to manage consumer organizations, see apic consumer-orgs.
   c. Select the required user registry.
   d. Specify the organization owner; complete one or other of the following steps, either to specify an existing user, or to create a new user:
      - To specify the user ID of an existing API Connect user who is in the selected user registry, complete the following steps:
        - For the Type of user, select Existing.
        - In the Username field, enter the user ID of the organization owner
        After the consumer organization is created, the specified user can immediately log in to the Developer Portal associated with the Catalog.
      - If you selected a Local User Registry, then to specify a new user ID, complete the following steps:
        - For the Type of user, select New User.
        - In the Username field, enter the new user ID.
        - Enter the Email, First Name, Last Name, and Password.
        After the consumer organization is created, the new user will have been added to the selected registry and can immediately log in to the Developer Portal associated with the Catalog, by using the specified user ID and password.
   e. Click Create to complete the creation of the consumer organization.

Editing a consumer organization

You can change the title of an IBM® API Connect consumer organization, and also transfer ownership to a new organization owner.

Before you begin

To complete this task, you must be defined as a member in a catalog or space (if spaces are enabled), and you must be assigned a role that has the Consumer-Org > Manage permission. For more information, see Adding provider organization users and assigning roles and Managing catalog membership.

About this task

If you transfer ownership, the new organization owner can be an existing user, or you can create a new user. When the ownership changes, the new owner is assigned administration privileges for the organization, and the previous owner has their administration privileges removed. If required, you can restore the privileges to the previous owner as described in Administering members and roles.

If Spaces are enabled in your Catalog, when you edit the details of a consumer organization, the updates are applied to the Catalog and its Spaces. For more information about enabling Spaces, see Using syndication in API Connect.

Procedure

To change the title of a consumer organization, or to transfer ownership to a new organization owner, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the name of the consumer organization that you want to work with.
5. To change the title, enter a new title in the Title field, then click Save.
6. To transfer ownership, select the user registry for the consumer organization owner you want to transfer ownership to.
   The remaining procedure varies according to the type of the selected user registry, as follows:
   - Local User Registry
     - Select whether the user is an Existing user or a New User.
       - For an existing user, complete the following steps:
         - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
         - Click Save.
       - For a new user, complete the following steps:
         - Enter a unique user name for the new user.
         - Supply an email address, name details, and a password.
         - Click Save.
   - LDAP
     - Enter the name of a user that exists in the selected user registry.
     - Click Save.
   - Authentication URL and OIDC
     - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account. If the required organization owner has not previously been invited and activated their account, you must ensure they have been invited to the organization first.
     - Click Save.

Results

If the title of the consumer organization is changed, all application developers that are members of that consumer organization will see the title change reflected in the Developer Portal user interface.

Related tasks

- Publishing a Product

Deleting a consumer organization

When you delete a consumer organization, all of the resources in that organization are removed from the cloud. However, the account of the organization owner remains in IBM® API Connect.

Before you begin

To complete this task, you must be defined as a member in a catalog or space (if spaces are enabled), and you must be assigned a role that has the Consumer-Org > Manage permission. For more information, see Adding provider organization users and assigning roles and Managing catalog membership.

Important: If you delete a consumer organization, client IDs and secrets associated with applications that were registered by users in the consumer organization can no longer be used to call APIs. However, the account of the organization owner remains in API Connect.

About this task

When you delete a consumer organization, it is removed permanently and users can no longer access that organization.
If Spaces are enabled in your Catalog, when you delete a consumer organization, it is deleted from the Catalog and its Spaces. For more information about enabling Spaces, see Using syndication in API Connect.

Procedure

To delete a consumer organization, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon ⋮ alongside the consumer organization you want to work with, then click Delete.

5. Click Confirm to delete the consumer organization.
   Note: If the consumer organization is the only one specified in the custom visibility settings for a published Product, the deletion operation will fail. For more information, see Changing the availability of a Product.

Working with the applications in a consumer organization

You can work with the developer applications that have been registered in a consumer organization.

Procedure

To work with applications, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon for the consumer organization that you want to work with, then click View Applications.
   A window opens listing all the developer applications that have been registered in the consumer organization
5. To view and edit the details of an application, click the title of the application.
6. Further options are available by clicking the options icon alongside the application you want to work with. For details, see Working with developer applications.

Working with the subscriptions in a consumer organization

You can work with the Plan subscriptions that have been created by the developer applications registered in a consumer organization.

Procedure

To work with Plan subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. Click the options icon for the consumer organization that you want to work with, then click View Subscriptions.
   A window opens listing all the Plan subscriptions that have been created by the developer applications registered in the consumer organization. If required, you can click the options icon alongside a subscription to work with that subscription. For details of the available options, see Working with application subscriptions.

Working with consumer organization groups

A *consumer organization group* is a collection of consumer organizations, and provides an efficient way of controlling who can see, and subscribe to, the APIs in your Products. By using a consumer organization group, you can define this access for all the developers in the organizations in that group in a single operation, rather than having to define access for the organizations separately.

About this task

A consumer organization group might represent particular business partners, internal organizations, or other groups of application developers.

When you create a Product, you can define which application developers can access the APIs in that Product. You can configure the following types of access:

- Visibility, which specifies the application developers who can see the APIs in the Developer Portal.
- Subscribability, which specifies the application developers who can create subscribe to use the APIs.

You configure this access by selecting the required consumer organizations, and Consumer organization groups. You can also change the visibility and subscribability settings for a Product after it has been published. For more information, see Creating a draft Product and Changing the availability of a Product.

Procedure

To work with consumer organization groups, you first navigate to the Consumer organization groups page. You then create new groups, send an email to the owners of the consumer organizations in a group, and edit or delete existing groups.

- Navigating to the required consumer organization groups page.
  1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
  2. If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
     a. Click the Spaces tab.

b. Select the space that you want to work with.
3. Click the Consumers tab, then click Manage Groups.

- Creating a consumer organization group.
  1. Click Add.
  2. Enter a Title for the new group and, optionally, a Summary. A Name is entered automatically. The title is used for display.
  3. Use the Search orgs/groups field to search for, and select, the consumer organizations that you want to add to the group.
     Note: To prevent excessive search results lists, only the first 10 matches are displayed when you begin typing in the field; lengthen your search string to shorten the list for locating the required consumer organization.
  4. Click Save to create the group.
- Editing a consumer organization group.
  1. Click the options icon ⋮ for the consumer organization group that you want to work with, then click Edit.
  2. Make the required updates, then click Save.

  Note: If you remove a consumer organization from a group, and that group is specified in the custom visibility settings for a Product, applications in that consumer organization can no longer subscribe to the Product. However, any existing subscriptions are not affected. For information on managing visibility and subscribability settings, see Editing a draft Product.
- Deleting a consumer organization group.
  1. Click the options icon ⋮ for the consumer organization group that you want to work with, then click Delete.
  2. Click Confirm to delete the group.
- Sending an email to the owners of the consumer organizations in a consumer group
  1. Click the options icon ⋮ for the consumer organization group that you want to work with, then click Message owners.
     The Send message window opens, and the email addresses of the consumer organization owners are displayed as the recipients.
  2. Enter the subject of the message in the Subject field.
  3. Enter the content of the message in the Message field, then click Send.

Sending messages to consumer organization owners

You can send email messages to the owners of consumer organizations. You can also style the email to match your business theme.

Before you begin

To complete this task, you must be defined as a member in a catalog or space (if spaces are enabled), and you must be assigned a role that has the Consumer-Org > Manage permission. For more information, see Adding provider organization users and assigning roles and Managing catalog membership.

Email messages contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see tbc.

Note: You can also send email messages by using the developer toolkit CLI, or by using the API Connect REST APIs. See API development and management commands, or API Connect REST APIs, for more information.

Procedure

To send a message, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.

   For more information about enabling spaces, see Using syndication in API Connect.
3. In the navigation pane of the API Manager UI, click the Consumers tab.
4. You can either send a message to the owner of a specific consumer organization, or to the owners of multiple selected consumer organizations.
   - To send a message to the owner of a specific consumer organization, click the options icon ⋮ alongside the consumer organization you want to work with, then click Message owner. The Send Message window opens, and the email address of the consumer organization owner is displayed as the recipient.
   - To send a message to the owners of multiple selected consumer organizations, complete the following steps:
     - Click the options icon ⋮ alongside the Add button, then click Message owners.
     - Select the required consumer organizations, then click Next. The Send Message window opens, and the email addresses of the selected consumer organization owners are displayed as the recipients.
5. Enter the subject of the message in the Subject field.
6. Select the Content type that you want to use for the entering the content of the message, from HTML, PlainText, or Both. The default content type is PlainText.
   An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
   For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 1. List of allowed HTML tags and their attributes

HTML tag	Attribute
`<a>`	`"class", "href", "hreflang", "style"`
`<em>`	`"class", "style"`
`<strong>`	`"class", "style"`
`<cite>`	`"class", "style"`
`<blockquote>`	`"class", "cite", "style"`
`<code>`	`"class", "style"`
`<ul>`	`"class", "type", "style"`
`<ol>`	`"class", "start", "type", "style"`
`<li>`	`"class", "style"`
`<dl>`	`"class", "style"`

HTML tag	Attribute
`<dt>`	`"class", "style"`
`<dd>`	`"class", "style"`
`<h1>`	`"class", "id", "style"`
`<h2>`	`"class", "id", "style"`
`<h3>`	`"class", "id", "style"`
`<h4>`	`"class", "id", "style"`
`<h5>`	`"class", "id", "style"`
`<h6>`	`"class", "id", "style"`
`<p>`	`"class", "style"`
`<div>`	`"class", "style"`
` `	`"class", "style"`
`<span>`	`"class", "style"`
`<img>`	`"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"`
`<table>`	`"class", "id", "style"`
`<tr>`	`"class", "id", "style"`
`<td>`	`"class", "id", "style"`

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.
Images can be used by adding a **`<img src="https://path/to/image.png"/>`** tag in the template. The **`src`** attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

7. Enter the content of the message.
8. Click Send.

Working with developer applications

You can work with the applications that have been registered in the Developer Portal associated with a Catalog.

About this task

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

Procedure

To work with developer applications, complete the following steps.

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. Click the Applications tab.
   All applications that have been registered in the Developer Portal associated with this Catalog are listed.
4. To view and edit the details of an application, click the title of the application.
5. Further options are available by clicking the options icon alongside the application you want to work with, as follows:
   - Client ID: lists the Client IDs for the application. You can add new client ID/client secret credentials; you can copy the client secret only at creation time. An option is provided for deleting any of the credentials from the application.
   - Create Subscription: subscribe the application to a Plan in a Product; select the required Plan and Product, then click Create Subscription.
   - View Subscriptions: a window opens listing summary details of all the Plan subscriptions for the application. If required, you can click the options icon alongside a subscription to work with that subscriptions. For details of the available options, see Working with application subscriptions.
   - View Consumer Organization: a window opens listing the consumer organization that the application belongs to. If required, you can click the options icon to work with the consumer organization. For details of the available options, see Working with consumer organizations.

Working with application subscriptions

You can work with the subscriptions to the Plans in all the Products in a specific Catalog. Plan subscriptions are created by application developers in the Developer Portal.

Procedure

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.
To work with application subscriptions, complete the following steps:

1. In the navigation pane of the API Manager UI, click Manage, then select the catalog that you want to work with.
   The Products page of the catalog opens, and all of the products available in that catalog are displayed.
2. Optional: If spaces are enabled in the catalog, select the space that you want to work with by completing the following steps:
   a. Click the Spaces tab.
   b. Select the space that you want to work with.
   For more information about enabling spaces, see Using syndication in API Connect.
3. Click the Subscriptions tab.
   A window opens listing all subscriptions to the Plans in all the Products in the Catalog.
4. Alongside the subscription that you want to work with, click the options icon ⋮.
   You can then select from the following options:
   - View Consumer Organization: a window opens listing the consumer organization that the subscribing application belongs to. If required, you can click the options icon ⋮ to work with the consumer organization For details of the available options, see Working with consumer organizations.
   - View Application: a window opens listing the subscribing application. If required, you can click the options icon ⋮ to work with the application. For details of the available options, see Working with developer applications.
   - View Plan: a window opens listing the Plan that the application is subscribed to. If required, you can click the options icon ⋮ to work with the Plan. For details of the available options, see Working with Plans.
   - View Product: a window opens listing the Product that contains the Plan that the application is subscribed to. If required, you can click the options icon ⋮ to work with the Product. For details of the available options, see Managing your Products.

Security and authentication

In API Manager, you can use TLS profiles to secure the transmission of data between the management server and other API Connect subsystems and external services, and also configure user registries to securely authenticate your Catalogs and APIs.

For details of authentication in IBM API Connect, see the following subtopics:

- **Creating a TLS client profile**
  In the IBM API Connect API Manager interface, TLS profiles are used to secure transmission of data between the management server and other API Connect subsystems and external services. TLS and SSL certificates guarantee that information you submit will not be stolen or tampered with. In this topic, you learn how to create a TLS profile in API Manager.
- **Authenticating by using your enterprise user registry**
  IBM API Connect supports a variety of user registry types for authenticating users and securing APIs.
- **Configuring a native OAuth provider**
  Native OAuth providers are configured and managed by you within your cloud.
- **Configuring a third-party OAuth provider**
  Enter the secure endpoints to provide OAuth authentication from a third party.
- **OAuth concepts for API Connect**
  OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

Creating a TLS client profile

In the IBM® API Connect API Manager interface, TLS profiles are used to secure transmission of data between the management server and other API Connect subsystems and external services. TLS and SSL certificates guarantee that information you submit will not be stolen or tampered with. In this topic, you learn how to create a TLS profile in API Manager.

Before you begin

One of the following roles is required to configure TLS Profiles:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

About this task

API Connect supports the use of TLS and SSL certificates, but does not itself produce strong encryption keys or manage your encryption keys. Encryption keys should be created and managed according to your own procedures. For more information, see Viewing certificate details and adding certificates to a keystore or truststore and Generating a PKCS#12 file for Certificate Authority.

Note: If you update a TLS profile that is associated with a Gateway service, the updates are not automatically propagated to Gateway servers. For instructions on configuring the toolkit command-line tool to use TLS certificates when connecting to API Manager, see Configuring the command-line tool to use TLS certificates.

Procedure

To create a TLS profile, complete the following steps:

1. In the API Manager, click Resources.
2. Select TLS.

3. Click Create in the TLS Client Profile table.
4. Enter the fields to configure the TLS Client Profile:

Field	Description
Title (required)	Enter a Title for the profile. The title is displayed on the screen.

Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands.
	To view the CLI commands to manage a TLS Client Profile, see apic tls-client-profiles.
	
	Important: The name of the TLS Client Profile as saved on the DataPower® Gateway, depending on the gateway type, is as follows:
	• DataPower API Gateway:
	`provider-org-name_catalog-name_tlsp-tls-profile-nameV1.0.0`
	• DataPower Gateway (v5 compatible):
	`provider-org-name-tls-profile-nameV1.0.0`
	
	where
	• *tls-profile-name* is the value of the auto-generated Name field for the TLS client profile in API Connect.
	• *provider-org-name* is the name of the provider organization containing the TLS client profile.
	• *catalog-name* is the name of the catalog, in that provider organization, containing the TLS client profile.

Version (required)	Assign a version number for the profile. Using version numbers allows you to create multiple server profiles with the same name and different configurations, for example, *MyProfile 1.0* and *MyProfile 1.1*.
Summary (optional)	Enter a description of the profile.
Protocols (required)	Select one or more supported TLS protocol versions. The default is 1.2.
Server Connection (optional)	Specify whether to support weak or insecure credentials. • Allow insecure server connections - Insecure server connections may result from self-signed certificates, expired or corrupted certificates, or certificates from an unknown or untrusted source. Check this box to allow the connection to proceed with an insecure connection. The default is to not allow insecure server connections. • Support Server Name Indication (SNI) - Check this box to enable SNI. SNI allows support for multiple certificates presented on the same IP address using different host names. The client profile sends the name of a virtual domain as part of the TLS negotiation. The default is to enable SNI.
Keystore (optional)	A Keystore is a repository containing public and private key pairs. Select the keystore where you will store the certificates for the profile. Default keystores are provided, and you can also create your own. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your uploaded certificates before they expire.
Truststore (optional)	A Truststore is a repository containing verified public keys, which are usually obtained from a third-party certificate authority. Truststores provide list of certificates to verify a peer's certificate. If used in a TLSServerProfile, a truststore is used when mutual authentication is enabled. Select a truststore for the profile. Default truststores are provided, and you can also create your own. Important: API Connect verifies certificates when you upload them, but does not continuously monitor them for expiry. You are responsible for monitoring and updating your uploaded certificates before they expire.
Ciphers (required)	Cipher suites are encryption/decryption algorithms used to secure HTTPs communication within the API Connect ecosystem. Select the ciphers that the profile supports. Note: The TLS 1.3 ciphers are clearly indicated. If you select TLS version 1.3 as one of the protocols for the profile but do not select any TLS 1.3 ciphers, **all** the TLS 1.3 ciphers are added to the list of ciphers supported by the profile. If you do not select TLS version 1.3 but select one or more TLS 1.3 ciphers, those ciphers are **not** added to the list of ciphers supported by the profile.

5. Click Save.

- **Creating a Keystore**
  Keystores contain matched pairs of public certificates and private keys used to confirm identity and encrypt/decrypt data transmission over HTTPS.
- **Creating a Truststore**
  Truststores are repositories containing trusted certificates with verified public keys. The certificates in the truststore are usually obtained from a third-party certificate authority (CA).
- **Generating a PKCS#12 file for Certificate Authority**
  PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.
- **Generating a self-signed certificate using OpenSSL**
  OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.
- **Viewing certificate details and adding certificates to a keystore or truststore**
  You can view details for the certificates in an existing keystore or truststore and add additional certificates. You might need add certificates when a certificate or PKCS #12 (P12) file expires.
- **Defining elliptic curve cryptographic schemes for a TLS client profile**
  You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

Related tasks

- Working with user registries

Creating a Keystore

Keystores contain matched pairs of public certificates and private keys used to confirm identity and encrypt/decrypt data transmission over HTTPS.

Before you begin

API Manager supports and uses TLS certificates, but does not produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see Generating a PKCS#12 file for Certificate Authority and Generating a self-signed certificate using OpenSSL.

One of the following roles is required to configure Keystores:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

About this task

API Connect includes pre-configured Keystores which may be used for testing purposes. For production environments, we suggest creating a new, secure Keystore.

Procedure

Perform the following steps to create a TLS Client profile:

1. In the API Manager, click Resources.
2. Select TLS.
3. Click Create in the Keystore table.

Field	Description
Title (required)	Enter a Title for the Keystore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage keystores, see apic keystores.
Summary (optional)	Enter a brief description.
Private Key & Public Key: Step 1: Upload private key	Upload the file containing the private key certificate. If necessary, you can click Browse to locate the file. If the file contains both the private and public keys, upload it in Step 1. Private and public keys are always uploaded in pairs, either in a single file or separate files.
Private key password (optional)	Enter the password for the private key if it has a password.
Private Key & Public Key: Step 2: Upload public key	If the public key is contained in a separate file, upload it in Step 2. Private and Public keys are always uploaded in pairs, either in a single file or separate files.

4. Click Save.
   Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

Related tasks

- Creating a TLS client profile
- Creating a Truststore
- Generating a PKCS#12 file for Certificate Authority

Creating a Truststore

Truststores are repositories containing trusted certificates with verified public keys. The certificates in the truststore are usually obtained from a third-party certificate authority (CA).

Before you begin

One of the following roles is required to configure truststores:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

About this task

API Manager supports and uses TLS certificates, but does not produce strong encryption keys or manage your encryption keys. Encryption keys are generated and managed according to your own procedures. For more information, see Generating a PKCS#12 file for Certificate Authority and Generating a self-signed certificate using OpenSSL.

API Connect includes pre-configured Truststores which may be used for testing purposes. For production environments, we suggest creating a new, secure Truststore.

Procedure

1. In the API Manager, click Resources.
2. Select TLS.
3. Click Create in the Truststore table.

Field	

Field	
Title (required)	Enter a Title for the Truststore. The title is displayed on the screen.
Name (required)	The Name is auto-generated. The value in the Name field is a single string that can be used in developer toolkit CLI commands. To view the CLI commands to manage truststores, see apic truststores.
Summary (optional)	Enter a brief description.
Public Keys	Upload the file containing the public key certificate. If necessary you can click Browse to locate the file.

4. Click Save.

Related tasks

- Creating a TLS client profile
- Creating a Keystore

Generating a PKCS#12 file for Certificate Authority

PKCS#12 (P12) files define an archive file format for storing cryptographic objects as a single file. API Connect supports the P12 file format for uploading a keystore and truststore. The keystore should contain both a private and public key along with intermediate CA certificates.

Before you begin

One of the following roles is required to add a key to a keystore or truststore:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

Before you can generate a P12 file, you must have a private key (for example: key.pem), a signed certificate by a Certificate Authority (for example certificate.pem) and one or more certificates from the CA authority (known as *intermediate CA certificates*).
Note: If your certificate file contains more than one certificate, you must manually split the file and create a single file for each entry. Each entry must be bound by the following markers:

```
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-------
```

Procedure

1. If you have intermediate certificates from your CA, concatenate them into a single .pem file to build your **caChain**. Be sure to enter a new line following each certificate's data.

```
cat ca1.pem ca2.pem ca3.pem > caChain.pem
cat caChain.pem
-----BEGIN CERTIFICATE-----
MIIEpjCCA46gAwIBAgIQEOd26KZabjd+BQMG1Dw16jANBgkqhkiG9w0BAQUFADCB
...
lQX7CkTJn6lAJUsyEa8H/gjVQnHp4VOLFR/dKgeVcCRvZF7Tt5AuiyHY
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEPDCCAySgAwIBAgIQSEus8arH1xND0aJ0NUmXJTANBgkqhkiG9w0BAQUFADBv
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIENjCCAx6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEwJTRTEU
...
-----END CERTIFICATE-----
```

2. Create the P12 file including the private key, the signed certificate and the CA file you created in step 1, if applicable. Omit the **-CAfile** option if you don't have CA certificates to include.
   The following command uses OpenSSL, an open source implementation of the SSL and TLS protocols.

```
openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12 -CAfile caChain.pem -chain
```

   Once the certificate file is created, it can be uploaded to a keystore.
3. Once the certificate file is created, it can be uploaded to a Truststore. In the API Manager, click .
4. Select TLS.
5. Click Create in the Keystore table.
6. Create a Keystore and upload the certificate file following the instructions at Creating a Keystore.
   Notes:
   - API Connect supports only the P12 (PKCS12) format file for the present certificate.
   - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
   - Your P12 file can contain a maximum of 10 intermediate certificates.
7. Click Save.

Results

Your P12 file is generated and ready to upload.

What to do next

Upload your P12 file to IBM® API Connect. For more information, see Creating a TLS client profile.

Generating a self-signed certificate using OpenSSL

OpenSSL is an open source implementation of the SSL and TLS protocols. It provides the transport layer security over the normal communications layer, allowing it to be intertwined with many network applications and services.

Before you begin

One of the following roles is required to complete this task:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

About this task

This topic tells you how to generate a self-signed SSL certificate request using the OpenSSL toolkit to enable HTTPS connections.

Procedure

To generate a self-signed SSL certificate using the OpenSSL, complete the following steps:

1. Write down the Common Name (CN) for your SSL Certificate. The CN is the fully qualified name for the system that uses the certificate. For static DNS, use the hostname or IP address set in your Gateway Cluster (for example. **`192.16.183.131`** or **`dp1.acme.com`**).
2. Run the following OpenSSL command to generate your private key and public certificate. Answer the questions and enter the Common Name when prompted.

   ```
   openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
   ```

3. Review the created certificate:

   ```
   openssl x509 -text -noout -in certificate.pem
   ```

4. Combine your key and certificate in a PKCS#12 (P12) bundle:

   ```
   openssl pkcs12 -inkey key.pem -in certificate.pem -export -out certificate.p12
   ```

5. Validate your P2 file.

   ```
   openssl pkcs12 -in certificate.p12 -noout -info
   ```

   Once the certificate file is created, it can be uploaded to a keystore.
6. In the API Manager, click Resources.
7. Select TLS.
8. Click Create in the Keystore table.
9. Create a Keystore and upload the certificate file following the instructions at Creating a Keystore.
   Note:
   - API Connect supports only the P12 (PKCS12) format file for the present certificate.
   - Your P12 file must contain the private key, the public certificate from the Certificate Authority, and all intermediate certificates used for signing.
   - Your P12 file can contain a maximum of 10 intermediate certificates.
10. Click Save.

Related tasks

- Creating a TLS client profile
- Creating a Keystore
- Creating a Truststore

Viewing certificate details and adding certificates to a keystore or truststore

You can view details for the certificates in an existing keystore or truststore and add additional certificates. You might need add certificates when a certificate or PKCS #12 (P12) file expires.

Before you begin

One of the following roles is required to edit Keystores and Truststores:

- Organization Administrator
- Owner
- Custom role with the `Settings: Manage` permissions

About this task

API Connect includes pre-configured Keystores and Truststores which can be used for testing purposes. For production environments, we suggest creating a new, secure Keystore or Truststore. From an existing Keystore or Truststore, you can edit the title, view the details of the certificates, and add new certificates.

Procedure

To work with existing Keystores and Truststores:

1. In the API Manager, click Resources.
2. Select TLS.
3. Click the name of the keystore or truststore.
4. Edit the name and summary if needed.
5. The Subject, Finger Print, and Expiration date is shown for each certificate in the keystore or truststore. Click > to view the certificate details.
6. Add additional private and/or public keys if needed.
7. Click Save.
   Note: After they have been uploaded, private keys cannot be downloaded from API Connect.

Related tasks

- Creating a Truststore
- Creating a Keystore
- Generating a PKCS#12 file for Certificate Authority
- Generating a self-signed certificate using OpenSSL

Defining elliptic curve cryptographic schemes for a TLS client profile

You define the elliptic curve cryptographic schemes for a TLS client profile by using the developer toolkit CLI.

About this task

To define elliptic curve cryptographic schemes for a TLS client profile, you include **elliptic_curve_auto_negotiation** and **elliptic_curve** properties in a YAML file definition for the TLS client profile. The **elliptic_curve** property lists the required elliptic curve cryptographic schemes. For example:

```
elliptic_curve_auto_negotiation: false
elliptic_curve:
  - secp521r1
  - secp384r1
  - prime256v1
```

You then use the developer toolkit CLI to create the TLS client profile in API Connect.

When **elliptic_curve_auto_negotiation** is set to **true**, the system negotiates the Elliptic-curve Diffie-Hellman (ECDH) key agreement automatically with its peer, and any **elliptic_curve** property settings are ignored.

The following example shows a complete YAML file for a TLS client profile:

```
type: tls_client_profile
name: my-tls-client-profile
version: 1.0.0
title: My TLS client profile
protocols:
  - tls_v1.2
ciphers:
  - ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
elliptic_curve_auto_negotiation: false
elliptic_curve:
  - sect163k1
insecure_server_connections: false
server_name_indication: true
```

Note:

- The **elliptic_curve_auto_negotiation** option is not supported by any of the API Connect gateway types. If the TLS client profile is targeted for an API Connect gateway; this setting is ignored by the gateway.
- The elliptic curve cryptographic schemes shown in each row of the following table are equivalent. However, API Connect recognizes only one or the other depending on how the TLS profile is used, as indicated in the table.

  Table 1.

API enforcement on the gateway	API Connect server access security
secp192r1	prime192v1
secp256r1	prime256v1

  Therefore, if you want to use either of these schemes and are unsure whether you are targeting the TLS client profile to the API Connect gateway for API enforcement, whether you are using it to secure user access to the API Connect servers, or whether it will be used for both purposes, specify both equivalent schemes; API Connect will simply ignore the non-relevant scheme. For example:

```
elliptic_curve:
        .
        .
        .
```

```
  - secp256r1
  - prime256v1
        .
        .
        .
```

Procedure

To create a TLS client profile with elliptic curve cryptographic schemes defined. complete the following steps:

1. Create a YAML file definition for your TLS client profile, with the required **elliptic_curve** property.
2. Log in to the management server from the developer toolkit CLI. Log in either as a member of the cloud administration organization or as a member of a provider organization, depending on where you want to create the TLS client profile. For details, see Logging in to a management server.
3. Create the TLS client profile by using the following command:

   ```
   apic tls-client-profiles:create --server mgmt_endpoint_url --org organization_name tls_client_profile_yaml_file
   ```

   where:
   - *mgmt_endpoint_url* is the platform API endpoint URL, and is the same as that which was used when you logged in at step 2.
   - *organization_name* is either **admin**, for the cloud administration organization, or the name of your provider organization, and is the same as that which was used when you logged in at step 2.
   - *tls_client_profile_yaml_file* is the name of the YAML file that contains the definition for your TLS client profile.

   Note: When you install IBM® API Connect, the API Connect gateway has a pre-supplied default TLS client profile that is used for API enforcement if you do not configure a TLS client profile; you cannot configure this default TLS client profile on the gateway.
   For reference details of all the **apic tls-client-profiles** commands, see apic tls-client-profiles.

   You can also complete the operations described in this topic by using the API Connect REST APIs; see the API Connect REST API documentation.

Authenticating by using your enterprise user registry

IBM® API Connect supports a variety of user registry types for authenticating users and securing APIs.

You can use your enterprise user registry for authentication in API Connect if it is of one of the following types:

LDAP directory
: If your user registry uses Lightweight Directory Access Protocol (LDAP), you can use it in API Connect for both user authentication and API security.

Authentication URL User Registry
: You can configure a non-LDAP user registry by using an authentication URL user registry. An authentication URL user registry enables integration with third-party authentication providers. You can use an authentication URL user registry in API Connect for both user authentication and API security.

Local User Registry
: You can authenticate users with a local user registry. A local user registry is an internal registry stored within API Connect.

OpenID Connect
: You can authenticate users through an identity provider that supports the OpenID Connect (OIDC) authentication protocol.

- **Working with user registries**
  To secure the APIs that are published to your IBM API Connect Catalogs, you authenticate with user registries.
- **Modifying the configuration details for a user registry**
  You modify the configuration details for a user registry by using the User Registries page in the API Manager user interface of IBM API Connect.
- **Password lockout criteria**
  You can be locked out of your account if you attempt to log in and fail consecutively.

Related tasks

- Working with user registries

Working with user registries

To secure the APIs that are published to your IBM® API Connect Catalogs, you authenticate with user registries.

About this task

Create user registry resources to secure your APIs, and authenticate with users.

Procedure

To configure a user registry, complete the following steps:

1. In the API Manager user interface, select Resources.
2. On the Resources page, select User Registries > Create.
3. Select the tile for the user registry type you want, and continue to the instructions for configuring user authentication for your selection.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the

Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

Authentication URL User Registry
: For more information, see Creating an Authentication URL user registry.

Custom user registry
: For more information, see Creating an organization-specific custom user registry.

LDAP User Registry
: For more information, see Creating an LDAP user registry in API Manager.

Local User Registry
: For more information, see Creating a Local User Registry

OIDC User Registry
: For more information, see Creating an OIDC user registry in API Manager.

Results

The user registry information is added to API Manager.

What to do next

You can now use API Manager to chart and manage your Catalogs.

- **Creating an Authentication URL user registry**
  An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.
- **Creating an organization-specific custom user registry**
  You can configure an organization-specific custom user registry to provide user authentication for the Developer Portal.
- **Creating an LDAP user registry in API Manager**
  You can use the API Manager UI to configure an organization-specific LDAP user registry to provide user authentication and onboarding for the Developer Portal. APIs can also be secured with an LDAP user registry.
- **Creating a Local User Registry**
  A Local User Registry (LUR) can be created to provide user authentication for API Manager.
- **Creating an OIDC user registry in API Manager**
  Create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

Creating an Authentication URL user registry

An Authentication URL user registry provides a simple mechanism for authenticating users by referencing a custom identity provider.

About this task

This topic describes how to create a new Authentication URL user registry as a Resource in your organization. After the user registry is created, the user registry must be added to the Sandbox catalog.

One of the following roles is required to configure user registries:

- Administrator
- Organization Owner
- Custom role with the `Settings: Manage` permissions

Note:
API Connect issues an HTTP **`GET`** call to the Authentication URL endpoint, sending the user's credential. The following example shows a call made to an Authentication URL identity provider with an endpoint defined as **`https://myauthurl.example.com/user/authenticate`**:

```
GET /user/authenticate HTTP/1.1
 Host: myauthurl.example.com
 Authorization: Basic c3Bvb246Zm9yaw=
```

If the Authentication URL endpoint returns an HTTP status code of **`200`**, the user authenticates successfully. An HTTP status code other than **`200`** indicates a failed login attempt. API Connect forwards any HTTP Header starting with **`X-`** (with the exception of **`X-Client-Certificate`**), and Cookie to the Authentication URL identity provider, to aid the authentication decision; for example:

```
GET /user/authenticate HTTP/1.1
 Host: myauthurl.example.com
 Authorization: Basic c3Bvb246Zm9yaw=
 X-Forwarded-For: 8.8.9.9
 X-Custom-Header-From-Customer: special
 Cookie: MyCookie=VGhpc0lzV2lja2VkQW1hemluZw==
```

When a user is presented with the form for completing their API Connect user registration, which fields are pre-populated depends on which fields are returned in the response from the Authentication URL identity provider. If any of the following fields are returned, they will be pre-populated in the registration form:

- **`username`**
- **`email`**
- **`first_name`**
- **`last_name`**

If the **`username`** field is not returned, the registration form displays the user name that was provided by the user. The pre-population capability requires that the response from the Authentication URL identity provider satisfies the following conditions:

- The **`Content-Type`** must be **`application/json`**.

- The response body format must be JSON.

A sample response is as follows:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "username":"myuser",
  "email":"myuser@example.com",
  "first_name":"My",
  "last_name":"User"
}
```

Procedure

1. In the API Manager, click Resources.
2. Select User Registries to see the list of current user registries in your organization.
3. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
4. Select Authentication URL User Registry and enter the following parameters:

Field	Description
Title (required)	Enter a descriptive name to use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Summary (optional)	Enter a brief description.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the **`Title`** of the User Registries when rendering them at the login page, rather than the **`Display Name`**.
URL (required)	Enter the URL for the authentication service. When establishing authentication, API Connect makes a GET call to the URL. The call includes the user name and password it has collected from the user in its authorization header. Either `200 OK` or `401 Unauthorized` will be returned.
TLS Client Profile (optional)	Select a TLS Client Profile to allow secure authentication with a specific web server.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend Authentication URL server: • **Only** select Case sensitive if your backend server supports case-sensitivity. • Do **not** select Case sensitive if your backend server does not support case-sensitivity. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.
6. Add the user registry to the Sandbox catalog. See Creating and configuring Catalogs.

Results

The user registry can be used for Basic Authentication in the Security Definition for an API. For more information, see Creating a basic authentication security definition.

Creating an organization-specific custom user registry

You can configure an organization-specific custom user registry to provide user authentication for the Developer Portal.

Before you begin

Custom user registries can be used for authenticating users to the Developer Portal, but cannot be used to secure APIs.

To configure a custom user registry as a resource in API Manager, the external user directory must be created and available to use with your API Connect ecosystem.

Also, a custom user registry definition for your external user registry must already have been created on the Cloud Manager, as you will need the information from this definition to set up the custom user registry. See Configuring a shared custom user registry for information.

One of the following roles is required to configure a custom user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create a custom user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure an organization-specific custom user registry. If you want to create a shared registry, see Configuring a shared custom user registry for more information.
Note:

- You can also create and manage custom user registries by using the API Connect REST APIs; see the API Connect REST API documentation.

You can use the following instructions to create a writable or a read-only custom user registry.

API Connect provides two methods for creating a custom user registry in the API Manager, as described in the following sections:

- Creating a custom user registry by using the API Manager UI
- Creating a custom user registry by using the developer toolkit CLI

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

Creating a custom user registry by using the API Manager UI

You configure a custom user registry by creating a new custom user registry resource, which references the custom user registry type that is defined in the Cloud Manager. To make the custom user registry available to the Developer Portal for user authentication, you must enable the registry in the associated Catalog. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the custom user registry type.

Creating a custom user registry resource

Use the following instructions to create a new custom user registry resource in the API Manager UI.

1. Click Create in the User Registries section of Resources.
2. Click the Custom user registry tile.
3. Enter the following information:

Property	Description
Custom type	The name of the custom user registry type. For example, *my-custom-user-registry-type.* Use the drop-down arrow to select different custom user registry types. The custom user registry type is defined in the Cloud Manager.
Title	A descriptive name to display in the UI.
Name	The name of the custom user registry. This name is auto-generated, and is used in CLI commands.
Summary	A brief description of the custom user registry.
Endpoint	Optional endpoint information.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend server: • **Only** select Case sensitive if your backend server supports case-sensitivity. • Do **not** select Case sensitive if your backend server does not support case-sensitivity. Note: • The Developer Portal does not support case sensitive usernames. • After at least one user has been onboarded into the registry, you cannot change this setting.
User registry managed	Determines whether API Connect manages your user registry. Valid values are: • **`true`** - select the checkbox • **`false`** - clear the checkbox
User managed	Determines whether your user registry is writable or not. Select the checkbox to set to **`true`** for writable. Clear the checkbox for the non-writable option.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Create. The new user registry is displayed in the User registries list.

Adding your custom user registry to the Developer Portal login

To make the custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. In the API Manager UI, click Manage followed by the relevant Catalog, and then click Catalog settings > Onboarding. In the Catalog User Registries section, click Edit, select the custom user registry, and click Save. For more information, see Creating and configuring Catalogs.

Creating a custom user registry by using the developer toolkit CLI

You configure an organization-specific custom user registry by creating a new custom user registry resource that references the integration document that exists on the Cloud Manager. You use developer toolkit CLI commands to create the custom user registry, and to make the registry available to the Developer Portal, you must enable the registry in the associated Catalog. When the custom user registry is used for authentication, API Connect makes a REST call to the endpoint of your external registry, as defined in the integration document.

Logging in to the management server CLI

Before you can define the custom user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the login command, see Logging in to a management server.

For more information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Defining your custom user registry

You define the configuration of your custom user registry in a *`custom_config_file`*`.yaml` file, as shown in the following example.

```
name: 'custom_registry_name'
title: 'display_title'
registry_type: 'custom_user_registry'
integration_url: custom_integration_url
case_sensitive: true_or_false
user_managed: true_or_false
user_registry_managed: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
 - name: provider_name
   title: 'provider_title'
configuration:
  custom_config1: 'value1'
  customize: true
```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
`name`	The name of the custom user registry. This name is used in the CLI commands.
`title`	A descriptive name to display in a graphical user interface.
`registry_type`	The registry type that is configured in the `name` property in the integration document, for example *`custom_user_registry`*.
`integration_url`	The custom integration URL in your API Connect configuration. You can determine the custom integration URL by using the following CLI command: `apic integrations:list --server mgmt_endpoint_url --subcollection user-registry`
`case_sensitive`	Determines whether your user registry is case-sensitive. Valid values are: • `true` • `false` To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** set `case_sensitive` to `true` if your backend LDAP server supports case-sensitivity. • Set `case_sensitive` to `false` if your backend LDAP server does not support case-sensitivity. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
`user_managed`	Determines whether your user registry is writable or not. Must be set to `true` for writable. Set to `false` if you don't want the registry to be writable.
`user_registry_managed`	Determines whether API Connect manages your user registry. Valid values are: • `true` • `false`

Property	Description
`email_required`	Determines whether an email address is required as part of the user onboarding process. Valid values are: • `true` • `false` If set to `true`, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
`email_unique_if_exist`	Determines whether email addresses must be unique within the user registry. Valid values are: • `true` • `false` Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
`identity_providers`	An array containing the details of your custom server, where: • `name` - is the name of the custom server and is the name that is used in CLI commands • `title` - is the display name of the custom server
`configuration`	The user-defined configuration based on the `configuration_schema` that is defined in the integration document.

Save your ***`custom_config_file`*`.yaml`** so it can be accessed by the **`user-registries:create`** command in the following section.

Creating your custom user registry

To create your organization-specific custom user registry, run the following CLI command:

`apic user-registries:create --server` ***`mgmt_endpoint_url`*** **`--org`** ***`organization_name custom_config_file`*****`.yaml`**

where:

- ***`mgmt_endpoint_url`*** is the platform API endpoint URL.
- **`--org`** ***`organization_name`*** means that the registry will be created in your provider organization.
- ***`custom_config_file`*** is the name of the YAML file that defines the configuration of your custom user registry.

On completion of the registry creation, the command displays the following summary details:

`registry_name registry_url`

The ***`registry_name`*** is derived from the **`name`** property in the custom user registry YAML file. The ***`registry_url`*** is the URL with which the custom registry resource can be accessed.
Your organization-specific user registry is now created; see the following section for instructions on how to make the registry available to users.

Configuring your custom user registry in a Catalog

To make your custom user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your custom user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   `apic user-registries:list --server` ***`mgmt_endpoint_url`*** **`--org`** ***`organization_name`***

2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   `apic configured-catalog-user-registries:create --server` ***`mgmt_endpoint_url`*** **`--org`** ***`organization_name`*** **`--catalog`** ***`catalog_name`*** **`-`**

   where *catalog_name* is the value of the `name` property of the required Catalog. The command returns

   `Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin`

3. Enter the following data, followed by a new line:

   `user_registry_url:` ***`custom_registry_url`***

   where ***`custom_registry_url`*** is the URL of your custom user registry, obtained in step 1.
4. Press **`CTRL D`** to terminate the input.

Related information

- Command-line tool reference for the developer toolkit

Creating an LDAP user registry in API Manager

You can use the API Manager UI to configure an organization-specific LDAP user registry to provide user authentication and onboarding for the Developer Portal. APIs can also be secured with an LDAP user registry.

Before you begin

To configure an LDAP user registry as a resource in API Manager, the LDAP directory must be created and populated for use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for securing a Catalog to authenticate Developer Portal users.

Important: If you are using an LDAP registry to secure APIs, the STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.
One of the following roles is required to configure an LDAP user registry:

- Organization Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used across the Cloud Manager, the API Manager, and the Developer Portal components in your environment.

This topic describes how to configure an organization-specific LDAP user registry. If you want to create a shared registry, see Configuring an LDAP user registry in the Cloud Manager for more information.
Note:

- You can also create and manage LDAP user registries by using the developer toolkit CLI. For more information, see Using the CLI to create an organization-specific LDAP user registry.
- The API Connect REST APIs can also be used to create and manage LDAP user registries; see the API Connect REST API documentation.
- You can map external LDAP groups to API Connect user roles to enable greater control of user access, but this configuration can be done only by using the developer toolkit CLI; see Using the CLI to create an organization-specific LDAP user registry for details.

You create an LDAP user registry by configuring a set of properties in the API Manager UI. If you want to enable writable LDAP, you must complete the Attribute Mapping section by selecting the User Managed checkbox, and providing the mapping of your source LDAP attribute names to the target API Connect values. You can also change a registry to be read-only again by clearing the User Managed checkbox. To make the registry available to the Developer Portal, you must define the registry for consumer onboarding in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions.

For general information about authenticating with LDAP, see LDAP authentication.

Procedure

Follow these steps to configure a new LDAP user registry as a Resource in the API Manager UI.

Note: If you are using an **`Active Directory`**, you must indicate this by using the property **`"directory_type": "ad"`** in the LDAP config.

1. In the API Manager, click Resources.
2. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
3. Select LDAP User Registry for the user registry type, and enter the following information:

Field	Description
Title	Enter a descriptive name to display on the screen.
Name	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the **`Title`** of the User Registries when rendering them at the login page, rather than the **`Display Name`**.
Summary (optional)	Enter a brief description.
Address	Enter the IP address or host name of the LDAP server.
Port	Enter the Port number that API Connect can use to communicate with the LDAP registry. For example, `389`.
Select a TLS Client Profile (optional)	Select the TLS Client Profile that the LDAP server requires.
Select a protocol version	Select the version number for the LDAP protocol that you are using.
Remote directory is Microsoft Active Directory	Select this option if you use Active Directory.
Case sensitive	To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** select Case sensitive if your backend LDAP server supports case-sensitivity. • Do **not** select Case sensitive if your backend LDAP server does not support case-sensitivity. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.

Field	Description
Unique email address	Select this checkbox if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

4. Click Next and enter the authentication information, which will vary depending on the selected Authentication Method. The choices are:
   - Compose DN - Select this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, **`uid=<username>,ou=People,dc=company,dc=com`** is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, Compose DN is not supported with the DataPower API Gateway.
   - Compose UPN - Select this format if your LDAP directory supports binding with User Principal Names such as **`john@acme.com`**. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.
     Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
   - Search DN - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.
     Optionally select a scope to specify which part of the directory information tree is examined:
     - Whole subtree (default)
     - Base object
     - Single level

   For all of the authentication methods:

   If you are creating an LDAP registry to authenticate users of an API, you can specify an LDAP authorization group to restrict API access. To be able to call an API that is secured by the LDAP registry, a user must successfully authenticate with their LDAP user ID and password **and** they must be a member of the specified authorization group. The authorization group can be a Static Group or Dynamic Group. A static group is one in which the individual members of the group are explicitly listed. A dynamic group is one which is defined according to the set of attributes that the group members share in common.

5. For authentication method Compose DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example `(uid=`.
Suffix	Specify the suffix to the DN. For example `)`.
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

6. For authentication method Compose UPN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Suffix	Enter the domain part of the user principal name. For example, `@acme.com`.
Use group authentication (optional)	Enter the Filter condition for the group.

7. For authentication method Search DN, enter the following:

Field	Description
Bind Method	Anonymous or Authenticated. If specific permissions are not needed to search the registry, select Anonymous Bind. Or, if specific permissions are necessary, select Authenticated Bind.
Admin DN	For Authenticated Bind, enter the Distinguished Name of a user authorized to perform searches in the LDAP directory. For example `cn=admin,dc=company,dc=com`.
Admin Password	For Authenticated Bind, enter the user password for the Admin DN.
Prefix	Specify the prefix to the DN. For example `(uid=`.
Suffix	Specify the suffix to the DN. For example `)`.
Base DN (optional)	Enter a base DN in the Base DN field, or click Get Base DN to populate the field with a retrieved base DN.
Use group authentication (optional)	Static or Dynamic. For Static Group, enter the Group Based DN, Prefix, and Suffix. For Dynamic Group, enter the Filter condition for the group.

8. Optional: Click Test configuration to test the settings for your LDAP user registry. Enter valid credentials to ensure that you can access the LDAP database.
9. Optional: If you want to make your LDAP user registry writable, select the User Managed checkbox in the Attribute Mapping section, and provide the mapping of your source LDAP attribute names to the target API Connect values. Click Add to add each name/value pair, specified as follows:
   - LDAP ATTRIBUTE NAME - is the name of the source LDAP attribute.
   - API CONNECT VALUE - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in **`[ ]`** with the value that the user supplies when signing up.

   The default user profile properties that API Connect requires during user registration are `username`, `first_name`, `last_name`, `email`, and `password`, as shown in the following example:

LDAP ATTRIBUTE NAME	API CONNECT VALUE
`dn`	`uid=[username],ou=users,dc=company,dc=com`
`cn`	`[first_name] [last_name]`
`sn`	`[last_name]`
`mail`	`[email]`
`userPassword`	`[password]`

You must ensure that you enter the correct attribute mapping values for your LDAP configuration, to enable API Connect to access the LDAP database. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.

10. Click Create.
    Your new LDAP registry is shown in the list of User Registries on the Resources page.

What to do next

If you want to make the LDAP user registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Click the relevant Catalog, then click Settings > Onboarding. In the Catalog User Registries section, click Edit, select the user registry, and click Save. For more information, see Creating and configuring catalogs .

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see Creating a basic authentication security definition.
- To use for authentication in the User Security configuration for a native OAuth provider, see Configuring user security for a native OAuth provider.

- **Using the CLI to create an organization-specific LDAP user registry**
  You can use the developer toolkit CLI to configure an organization-specific LDAP user registry to provide user authentication for the Developer Portal. APIs can also be secured with an LDAP user registry.

Related information

- Command-line tool reference for the developer toolkit
- Securing your API Connect Cloud with LDAP (series of developer articles)

Using the CLI to create an organization-specific LDAP user registry

You can use the developer toolkit CLI to configure an organization-specific LDAP user registry to provide user authentication for the Developer Portal. APIs can also be secured with an LDAP user registry.

Before you begin

To configure an LDAP user registry as a resource in API Manager, the LDAP directory must be created and available to use with your API Connect ecosystem.

LDAP registries can be used to secure APIs, or for securing a Catalog to authenticate Developer Portal users.

Important: If you are using an LDAP registry to secure APIs, The STARTTLS protocol, which upgrades an insecure protocol to a secure one by applying TLS security, is not supported.
Note: If you are using an **`Active Directory`**, you must indicate this by using the property **`"directory_type": "ad"`** in the LDAP config.
One of the following roles is required to configure an LDAP user registry:

- Administrator
- Owner
- Topology Administrator
- Custom role with the `Settings: Manage` permissions

About this task

You can create an LDAP user registry that is specific to a provider organization, or one that can be shared and available to all of the provider organizations in your API Connect environment. An organization-specific LDAP user registry can be used for onboarding and authenticating Developer Portal users in a specific provider organization. While a shared LDAP user registry can be used for authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to configure an organization-specific LDAP user registry. If you want to create a shared registry, see Using the CLI to configure a shared LDAP user registry for more information.
Note:

- You can also create organization-specific LDAP user registries by using the API Manager UI; for more information see Creating an LDAP user registry in API Manager.
- In addition, you can create and manage LDAP user registries by using the API Connect REST APIs; see the API Connect REST API documentation.
- If you are using the DataPower® API Gateway, LDAP group authentication is not supported.
- You can map external LDAP groups to API Connect user roles to enable more control of user authorization by using the developer toolkit CLI. The following instructions explain how to set the **`external_group_mapping_enabled`** configuration on your LDAP user registry resource. For information about how to set external role mapping on your API Connect user roles, see Configuring LDAP group mapping on Cloud Manager user roles.

You create an LDAP user registry by first defining the registry details in a configuration file. You then use a developer toolkit CLI command to create the registry, passing the configuration file as a parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated Catalog. To secure APIs with an LDAP registry, you must configure security definitions. You can use the following instructions to create a writable or a read-only LDAP user registry.

For general information about authenticating with LDAP, see LDAP authentication.

Logging in to the management server CLI

Before you can define the LDAP user registry configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the login command, see Logging in to a management server.

For more information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Defining your LDAP configuration

You define the configuration of your LDAP user registry in an *`ldap_config_file`*`.yaml` file, as shown in the following example. Note that the actual contents of your YAML file will vary depending on the authentication method of your LDAP server, and this is explained in the following tables.
Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.

```
name: registry_name
title: "display_title"
integration_url: LDAP_integration_url
user_managed: true_or_false
user_registry_managed: false
directory_type: ad
external_group_mapping_enabled: true_or_false
case_sensitive: true_or_false
email_required: true_or_false
email_unique_if_exist: true_or_false
identity_providers:
 - name: provider_name
   title: provider_title
endpoint:
  endpoint: "ldap_server_url_and_port"
configuration:
  authentication_method: authentication_method
  authenticated_bind: "true_or_false"
  admin_dn: "admin_dn"
  admin_password: admin_password
  search_dn_base: "search_dn_base"
  search_dn_scope: search_dn_scope
  search_dn_filter_prefix: prefix
  search_dn_filter_suffix: suffix
  attribute_mapping:
    dn: "distinguished_name"
    cn: "common_name"
    sn: "last_name"
    mail: "email_address"
    userPassword: "password"
```

The registry properties that are common to each authentication method are described in the following table:

Property	Description
`name`	The name of the registry. This name is used in CLI commands.
`title`	A descriptive name to display in a graphical user interface.
`integration_url`	The LDAP integration URL in your API Connect configuration. You can determine the LDAP integration URL by using the following CLI command: `apic integrations:list --server mgmt_endpoint_url --subcollection user-registry`
`user_managed`	Determines whether your user registry is writable or not. Must be set to `true` for writable LDAP. You can change this setting to `false` if you don't want the registry to be writable; see the Switching your LDAP registry between writable and read-only section at the end of this topic for details. Note that a writable LDAP user registry cannot be used to authenticate Cloud Manager and API Manager users.
`user_registry_managed`	Must be set to `false` for LDAP. Determines whether API Connect manages your user registry. Only LUR registries are managed by API Connect.
`directory`	Specify one of the of following values for the directory type: • `ad` - Indicates that the LDAP is Microsoft Active Directory • `standard` - (Default value) Indicates any other LDAP
`external_group_mapping_enabled`	Determines whether your user registry supports LDAP group mapping. Valid values are: • `true` • `false` The default value is `false`.

Property	Description
`case_sensitive`	Determines whether your user registry is case-sensitive. Valid values are: • `true` • `false` To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on your backend LDAP server: • **Only** set `case_sensitive` to `true` if your backend LDAP server supports case-sensitivity. • Set `case_sensitive` to `false` if your backend LDAP server does not support case-sensitivity. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
`email_required`	Determines whether an email address is required as part of the user onboarding process. Valid values are: • `true` • `false` If set to `true`, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
`email_unique_if_exist`	Determines whether email addresses must be unique within the user registry. Valid values are: • `true` • `false` Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
`identity_providers`	An array containing the details of your LDAP server, where: • `name` - is the name of the LDAP server and is the name that is used in CLI commands • `title` - is the display name of the LDAP server
`endpoint`	The endpoint of your LDAP server, made up of the url and port, for example: `"ldap://server.com:389"`
`tls_profile`	Optionally set the TLS Client Profile that the LDAP server requires.
`protocol_version`	Optionally set the version number for the LDAP protocol that you are using. Valid values are: • `2` • `3` Defaults to `3` if not explicitly set.

The properties in the configuration section will vary depending on the selected authentication method. The three authentication methods are:

- `compose_dn` - Set this format if you can compose the user LDAP Distinguished Name (DN) from the user name. For example, `uid=<username>,ou=People,dc=company,dc=com` is a DN format that can be composed from the user name. If you are unsure whether Compose (DN) is the correct option, contact your LDAP administrator. If you are using an LDAP registry to secure APIs, `compose_dn` is not supported with the DataPower API Gateway.
- `compose_upn` - Set this format if your LDAP directory supports binding with User Principal Names such as `john@acme.com`. The Microsoft Active Directory is an example of an LDAP directory that supports Compose UPN authentication. If you are unsure whether your LDAP directory supports binding with UPNs, contact your LDAP administrator.
  Note: The Admin Bind DN and Admin Bind Password are not used with this authentication method.
- `search_dn` - Select this format if you cannot compose the user LDAP Distinguished Name from the user name; for example, if the base DNs of the users are different. This format might require an administrator DN and password to search for users in the LDAP directory. If your LDAP directory permits anonymous binds, you can omit the admin DN and password. If you are unsure if your LDAP directory permits anonymous binds, contact your LDAP administrator.

For authentication method `compose_dn`, set the following configuration properties:

Properties	Description
`authentication_method`	`compose_dn`
`authenticated_bind`	The bind method. Valid values are: • `"true"` - authenticated bind • `"false"` - anonymous bind If specific permissions are not needed to search the registry, select `"false"`. If specific permissions are necessary, select `"true"`.
`admin_dn`	If `authenticated_bind` is set to `"true"`, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: `"cn=admin,dc=company,dc=com"`
`admin_password`	If `authenticated_bind` is set to `"true"`, enter the user password for the `admin_dn`.
`search_dn_base`	Optionally set a base DN, for example: `"dc=company,dc=com"`
`bind_prefix`	Set the prefix to the DN, for example: `(uid=`
`bind_suffix`	Set the suffix to the DN, for example: `)`

Properties	Description
attribute_mapping	If **user_managed** is set to **true**, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows: ***ldap_registry_attribute_name*: "*apic_ldap_attribute_value*"** Where: • *ldap_registry_attribute_name* - is the name of the source LDAP attribute • *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in **[ ]** with the value that the user supplies when signing up. The user profile properties that API Connect requires during user registration are **username**, **first_name**, **last_name**, **email**, and **password**. The following extract shows an example of an attribute mapping: **attribute_mapping:** **dn: "uid=[username],ou=users,dc=company,dc=com"** **cn: "[first_name] [last_name]"** **sn: "[last_name]"** **mail: "[email]"** **userPassword: "[password]"**

For authentication method **compose_upn**, set the following configuration properties:

Properties	Description
authentication_method	**compose_upn**
authenticated_bind	The bind method. Valid values are: • **"true"** - authenticated bind • **"false"** - anonymous bind If specific permissions are not needed to search the registry, select **"false"**. If specific permissions are necessary, select **"true"**.
admin_dn	If **authenticated_bind** is set to **"true"**, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: **"cn=admin,dc=company,dc=com"**
admin_password	If **authenticated_bind** is set to **"true"**, enter the user password for the **admin_dn**.
bind_suffix	Enter the domain part of the user principal name. For example: **@acme.com**
attribute_mapping	If **user_managed** is set to **true**, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows: ***ldap_registry_attribute_name*: "*apic_ldap_attribute_value*"** Where: • *ldap_registry_attribute_name* - is the name of the source LDAP attribute • *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in **[ ]** with the value that the user supplies when signing up. The user profile properties that API Connect requires during user registration are **username**, **first_name**, **last_name**, **email**, and **password**. The following extract shows an example of an attribute mapping: **attribute_mapping:** **dn: "uid=[username],ou=users,dc=company,dc=com"** **cn: "[first_name] [last_name]"** **sn: "[last_name]"** **mail: "[email]"** **userPassword: "[password]"**

For authentication method **search_dn**, set the following configuration properties:

Property	Description
authentication_method	**search_dn**
authenticated_bind	The bind method. Valid values are: • **"true"** - authenticated bind • **"false"** - anonymous bind If specific permissions are not needed to search the registry, select **"false"**. If specific permissions are necessary, select **"true"**.
admin_dn	If **authenticated_bind** is set to **"true"**, enter the Distinguished Name (DN) of a user authorized to perform searches in the LDAP directory. For example: **"cn=admin,dc=company,dc=com"**
admin_password	If **authenticated_bind** is set to **"true"**, enter the user password for the **admin_dn**.
search_dn_base	Optionally set a base DN, for example: **"dc=company,dc=com"**
search_dn_scope	Optionally set the search DN scope. The scope determines which part of the directory information tree is examined. Possible values are: • **base** • **one** • **sub** (default value)

Property	Description
search_dn_filter_prefix	Set the prefix to the DN, for example: **(uid=**
search_dn_filter_suffix	Set the suffix to the DN, for example: **)**
attribute_mapping	If **user_managed** is set to **true**, provide the mapping of your source LDAP attribute names to the target API Connect values. This mapping is configured as a name/value pair, specified as follows: ***ldap_registry_attribute_name*: "*apic_ldap_attribute_value*"** Where: • *ldap_registry_attribute_name* - is the name of the source LDAP attribute • *apic_ldap_attribute_value* - is a string that represents the value that API Connect will populate the LDAP attribute with, by replacing the content contained in **[ ]** with the value that the user supplies when signing up. The user profile properties that API Connect requires during user registration are **username**, **first_name**, **last_name**, **email**, and **password**. The following extract shows an example of an attribute mapping: **attribute_mapping:** **dn: "uid=[username],ou=users,dc=company,dc=com"** **cn: "[first_name] [last_name]"** **sn: "[last_name]"** **mail: "[email]"** **userPassword: "[password]"**

Save your ***ldap_config_file*.yaml** so it can be accessed by the **user-registries:create** command in the following section. See the Example section for an example configuration file.

Creating your LDAP user registry

To create your organization-specific LDAP user registry, run the following CLI command:

apic user-registries:create --server *mgmt_endpoint_url* --org *organization_name* *ldap_config_file*.yaml

where:

- *mgmt_endpoint_url* is the platform API endpoint URL.
- *organization_name* is the value of the **name** property of your provider organization.
- *ldap_config_file* is the name of the YAML file that defines the configuration of your LDAP user registry.

On completion of the registry creation, the command displays the following summary details:

registry_name registry_url

The *registry_name* is derived from the **name** property in the configuration YAML file. The *registry_url* is the URL with which the registry resource can be accessed. Your LDAP user registry is now created; see the following section for instructions on how to make the registry available in the Developer Portal.

Configuring your LDAP registry in a Catalog

If you want to make your LDAP registry available for authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your LDAP user registry by using the following command (or you can copy and paste from the summary of the registry creation):

   apic user-registries:list --server *mgmt_endpoint_url* --org *organization_name*

2. Log in to the management server as a member of a provider organization; enter the following command:

   apic login --server *mgmt_endpoint_url* --username *user_id* --password *password* --realm provider/*identity_provider*

   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   apic identity-providers:list --scope provider --server *mgmt_endpoint_url* --fields title,realm

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
   For full details of the **apic login** command, see Logging in to a management server.

3. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   apic configured-catalog-user-registries:create --server *mgmt_endpoint_url* --org *organization_name* --catalog *catalog_name* -

   where *catalog_name* is the value of the `name` property of the required Catalog. The command returns

```
Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
```

4. Enter the following data, followed by a new line:

   ```
   user_registry_url: ldap_registry_url
   ```

   where *ldap_registry_url* is the URL of your LDAP registry, obtained in step 1.
5. Press **CTRL D** to terminate the input.

Switching your LDAP registry between writable and read-only

After an LDAP user registry has been created, it can be switched between writable and read-only by updating the **user_managed** property in the registry configuration. Complete the following steps.

1. Determine the name or ID of the LDAP user registry that you want to update, by running the following command (or you can use the summary from the registry creation):

   ```
   apic user-registries:list --server mgmt_endpoint_url --org organization_name
   ```

   The command returns a list of all the user registries for that organization, shown by name followed by their registry URL. The registry ID is located at the end of the URL, for example **https://company.com/api/user-registries/x-x-x-x-x/*registry_id***.
2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic user-registries:update --server mgmt_endpoint_url --org organization_name registry_name_or_id -
   ```

   where ***registry_name_or_id*** is the name or ID of the LDAP user registry that you want to update (as determined in the previous step). The command returns:

   ```
   Reading USER_REGISTRY_FILE arg from stdin
   ```

3. Enter the following data, followed by a new line:

   ```
   user_managed: true_or_false
   ```

   where **true** makes the registry writable, and **false** makes the registry read-only.
4. Press **CTRL D** to terminate the input.

Note that if you are changing your registry from read-only to writable, you must also set the **attribute_mapping** configuration, as described in the previous registry property tables.

Using an LDAP user registry to secure APIs

If you want to use the LDAP user registry to secure APIs, see the following information:

- To use for basic authentication in the security definition for an API, see Creating a basic authentication security definition.
- To use for authentication in the User Security configuration for a native OAuth provider, see Configuring user security for a native OAuth provider.

For details of all the **apic user-registries** and **apic configured-catalog-user-registries** commands, see apic user-registries and apic configured-catalog-user-registries.

Example

The following example shows a configuration file that uses the Search DN authentication method for setting up writable LDAP:

```
name: sdn-ldap
title: "SDN LDAP User Registry"
integration_url: https://mycompany.com/api/cloud/integrations/user-registry/xxx-xxx-xxx
user_managed: true
user_registry_managed: false
directory_type: standard
case_sensitive: false
identity_providers:
  - name: ldap
    title: "SDN LDAP Identity Provider"
endpoint:
  endpoint: "ldap://mycompany.com:389"
configuration:
  authentication_method: search_dn
  authenticated_bind: "true"
  admin_dn: "cn=admin,dc=company,dc=com"
  admin_password: xxxx
  search_dn_base: "dc=company,dc=com"
  search_dn_scope: sub
  search_dn_filter_prefix: (uid=
  search_dn_filter_suffix: )
  attribute_mapping:
    dn: "uid=[username],ou=users,dc=company,dc=com"
    cn: "[first_name] [last_name]"
    sn: "[last_name]"
    mail: "[email]"
    userPassword: "[password]"
```

Related information

- Command-line tool reference for the developer toolkit
- Securing your API Connect Cloud with LDAP (series of developer articles)

Creating a Local User Registry

A Local User Registry (LUR) can be created to provide user authentication for API Manager.

About this task

Local User Registries (LURs) are the default user registries included in API Connect. LURs are local databases included with API Connect. Two default LURs are installed and configured during installation of API Connect. They cannot be deleted. The default Admin user account is stored in the Provider LUR.

You can use API Manager to create additional Local User Registries for use with your provider organization.

One of the following roles is required to configure user registries:

- Administrator
- Organization Owner
- Custom role with the `Settings: Manage` permissions

Procedure

Follow these steps to configure a new LUR:

1. In the API Manager, click Resources.
2. Select User Registries to see the list of current user registries in your organization.
3. Click Create in the User Registries section.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
4. Select Local User Registry as the type for the user registry and enter the following information:

Field	Description
Title (required)	Enter a descriptive name for use on the screen.
Name (required)	The name that is used in CLI commands. The name is auto-generated. For details of the CLI commands for managing user registries, see apic user-registries.
Display Name (required)	The name that is displayed for selection by the user when logging in to a user interface, or activating their API Manager account. For details of user interface log in, and account activation, see Accessing the Cloud Manager user interface, Accessing the API Manager user interface, and Activating your API Manager user account. Note: The Developer Portal uses the **`Title`** of the User Registries when rendering them at the login page, rather than the **`Display Name`**.
Summary (optional)	Enter a brief description.
Case sensitive	Select this setting if user names are case-sensitive. Note: The Developer Portal does not support case sensitive usernames. Note: After at least one user has been onboarded into the registry, you cannot change this setting.
Email required	Select this checkbox if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this checkbox if email addresses must be unique within the user registry. For new Local User Registries, this setting is always selected; so if email addresses are contained in the user record, they must be unique. However, for existing Local User Registries this setting can be edited. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

5. Click Save.
6. Add the user registry to the Sandbox Catalog. See Creating and configuring Catalogs.

Results

The user registry can be used for Basic Authentication in the Security Definition for an API. For more information, see Creating a basic authentication security definition.

Creating an OIDC user registry in API Manager

Create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

You can create an OIDC user registry that is specific to a provider organization, or that is shared and available to all the provider organizations in your API Connect environment. An organization-specific OIDC user registry is used for onboarding and authenticating Developer Portal users, while a shared OIDC user registry can be used for onboarding and authenticating Cloud Manager, API Manager, and Developer Portal users.

This topic describes how to create an organization-specific registry. For information on how to create a shared registry, see Configuring an OIDC user registry in Cloud Manager.

Important: If you are using X (formerly known as Twitter) as your OIDC provider, API Connect supports only the OAuth 1.0a method, so your X application must be configured to use OAuth 1.0a. Other OIDC providers support OAuth 2.0.

API Connect provides two methods for creating an OIDC user registry in API Designer, as described in the following sections:

- Using the UI to create an OIDC user registry
- Using the CLI to create an OIDC user registry

Using the UI to create an OIDC user registry

Use the API Designer user interface to create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

1. In the API Designer navigation pane, click Resources.
2. Click User Registries.
3. Click Create and select OIDC User Registry.
   Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access.
4. On the Create OIDC User Registry page, use the fields in each of the following sections to configure the registry settings, and then click Create.
   Many of the registry settings are preconfigured to simplify the configuration steps.

   Provider Type
   Use the settings in Table 1 to define the provider type.

   Table 1. Provider Type settings

Field	Description
Provider Type	OIDC provider. Select one of the following supported OIDC providers: • Facebook • GitHub • Google • LinkedIn • Slack • Twitter • Windows Live • Standard OIDC (default value allows you to specify another provider)
Title	Provide a descriptive name for display purposes.
Name	Automatically generated. This name is used in CLI commands to reference the registry. For details of the CLI commands for managing user registries, see the apic user-registries topic in the Command Line tool reference section of this documentation.
Summary	Provide a brief description of the new registry.
Email required	Select this check box if an email address is required as part of the user onboarding process. If selected, the source identity provider must supply the email address as part of the authentication process during onboarding. Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
Unique email address	Select this check box if email addresses must be unique within the user registry. Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.

   Provider Endpoint
   Automatically generated for most supported providers. In the Authorization Endpoint field, type the URL of the provider's authorization endpoint.
   Token Endpoint
   Fill in the settings as described in Table 2.

   Table 2. Token Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's token endpoint.
TLS	Select the TLS Client Profile for the token endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

   UserInfo Endpoint
   Fill in the settings as described in Table 3.

   Table 3. UserInfo Endpoint settings

Field	Description
URL	Preconfigured for most of the supported OIDC providers. Type the URL of the provider's userinfo endpoint. For LinkedIn, the endpoint changed and now uses the following value: `https://api.linkedin.com/v2/me` If you previously configure the OIDC registry using the legacy LinkedIn OIDC protocol, you do not need to specify this endpoint.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

   Email Endpoint
   Required for the legacy LinkedIn OIDC protocol.
   Due to changes in LinkedIn, the configuration settings for the OIDC registry in API Connect also changed. If you are configuring a new OIDC registry for LinkedIn, you can leave the Email Endpoint field blank.

   If you previously configured an OIDC registry for use with the legacy LinkedIn OIDC protocol, update your configuration by providing the Email endpoint settings as described in Table 4. The new setting is required for the legacy LinkedIn OIDC protocol.

   Table 4. Email Endpoint settings

Field	Description

Field	Description
URL	For the legacy LinkedIn OIDC protocol, this field defaults to the following value: `https://api.linkedin.com/v2/emailAddress?q=members&projection=(elements*(handle~))` You can override the setting by typing in a new value.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

JWKS Endpoint

Fill in the settings as described in Table 5.

Table 5. JWKS Endpoint settings

Field	Description
URL	Type the URL of the read-only endpoint that contains the public keys' information in JWKS format.
TLS	Select the TLS Client Profile for the userinfo endpoint (OIDC must be configured to use TLS). Default is Default TLS Client Profile.

Logout

Fill in the settings as described in Table 6.

Table 6. Logout

Field	Description
Logout redirect	Optionally, give a logout redirect URL, allowing APIC to initiate an https 302 redirect to the configured endpoint. Note: When the Developer Portal uses this feature, it does not retain control afterwards. Instead, the content is presented from the redirected endpoint.

Client Information

Fill in the settings as described in Table 7.

Table 7. Client Information settings

Field	Description
Client ID	Provide the client ID of the application that is registered with the selected OIDC provider.
Client Secret	Provide the client secret of the application that is registered with the selected OIDC provider. This field is optional when you create an OIDC directory of type Standard and set the Client Authentication Method to Data encoded form body. If you do not use the Client Secret, make sure that mutual TLS is possible with the OIDC provider.
Response Type	Preconfigured for most of the supported OIDC providers. Specify the data type of the response that will be received from the OIDC provider.
Scopes	Preconfigured for most of the supported OIDC providers. Specify the access scope for the OIDC provider.
Client Authentication Method	Preconfigured for most of the supported OIDC providers. Select the authentication method to be used with the OIDC provider. Options are: • HTTP Basic authentication schema • Data encoded form body

Additional Support

Optional. Select the additional security parameters described in Table 8.

Table 8. Additional security options

Security parameter	Setting name for CLI	Description
NONCE	`nonce`	Enable the NONCE extension to prevent compromised requests from being used again (replayed).
Proof Key for Code Exchange (PKCE)	`pkce`	Enable the PKCE extension to allow public clients to mitigate the threat of having the authorization code intercepted.

Advanced Features

Optional. Select the advanced features described in Table 9.

Table 9. Advanced features

Feature / UI Label	Setting name for CLI	Description
Auto onboard	`auto_onboard`	Allow users to executes calls to APIs without logging in first, provided they present a valid token issued by the OIDC provider. Note: Does not apply to consumer organizations.
Always use the userinfo endpoint	`userinfo`	Configures the OIDC user registry to always fetch user data from the userinfo endpoint, if populated.
Use Portal as Endpoint for external OIDC provider traffic	`proxy_redirect`	When authenticating users, redirect the external OIDC provider to communicate with the Developer Portal instead of API Manager.
Return third-party access token	`proxy_access_token`	Include the third-party OIDC access token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an ERR_HTTP2_PROTOCOL_ERROR or ERR_CONNECTION_CLOSED error.
Return third-party id_token	`proxy_id_token`	Include the third-party OIDC id_token in the response. Note: Enable this setting for debugging purposes only. This setting is not recommended for use in a production environment. When this setting is enabled, the token size will increase when a request is made to API Connect using the token. The larger token size might exceed the HTTP protocol limit, resulting in an ERR_HTTP2_PROTOCOL_ERROR or ERR_CONNECTION_CLOSED error.
Token Relay	`token_relay`	Allow `access_token/refresh_token` to send in 302 redirect for logout
Userinfo POST	`post_userinfo`	If supported by your OIDC provider, by using HTTP POST method when contacting userinfo endpoint. Note: Not all OIDC providers support POST. Ensure that your OIDC provider supports this before you enable the feature.

Feature / UI Label	Setting name for CLI	Description
Use IBM APIC token expiration setting from cloud	`override_provider_ttl`	Override the OIDC provider's token expiration setting with the access token and refresh token expiration settings that are configured in API Connect. For information on configuring the access token and refresh token expiration settings in API Connect, see Configuring timeouts for access tokens and refresh tokens.
Disable hash verification (CLI only)	`disable_hash_verification`	Disable `at_hash`, `c_hash`

User Mapping

Fill in the settings as described in Table 10.

Note: The User Mapping fields are preconfigured for most of the supported OIDC providers to minimize potential errors; use care when changing the settings. For the Standard OIDC option, contact your OIDC provider to obtain the details of the fields.

Table 10. User mapping settings

Field	Description
Username	The name of the field in the response token that contains the user's user name. Note: The Username field must be unique for this OIDC registry, because it identifies the user in the system and cannot be changed.
Email	The name of the field in the response token that contains the user's email address.
First name	The name of the field in the response token that contains the user's given name.
Last name	The name of the field in the response token that contains the user's surname.

Redirect URI

The Redirect URI section lists the endpoints to which the OIDC authorization server will redirect the authorization code. There is one endpoint for each API Connect organization type: Cloud administration, Provider organization, and Consumer organization. The redirect endpoint is required when a user registers their application with the OIDC provider. For example, if this OIDC user registry is used by a provider organization, the user must register the provider organization's redirect endpoint with the OIDC provider. These read-only fields are provided for you to copy the endpoint values as required.

Enabling the OIDC user registry

Complete the following steps to enable the new user registry for a specific catalog in the Developer Portal. Repeat this procedure for each catalog that will use the new registry.

1. On the navigation pane, click Manage.
2. Select the catalog for which you will enable the new OIDC user registry.
3. On the navigation pane, click the Catalog settings tab.
4. On the Settings page, click Onboarding.
5. On the Onboarding page, click Edit in the Catalog User Registries section.
6. On the Edit User Registry page, select the new OIDC user registry to enable for your catalog.
7. Click Save.

Using the CLI to create an OIDC user registry

Use the developer toolkit CLI to create an organization-specific OIDC user registry when multi-factor authentication (MFA) is required.

Important: Do not share user registries between the API Manager and the Developer Portal, or between Developer Portal sites when self-service onboarding is enabled or account deletions in any of the sites are expected. You should create separate user registries for them, even if the separate registries point to the same backend authentication provider (for example, an LDAP server). This separation enables the Developer Portal to maintain unique email addresses across the Catalog, without API Manager needing the same requirement. It also avoids problems with users deleting their accounts from the Developer Portal that then affects their API Manager access. Configure an OIDC user registry by first defining the registry details in a configuration file. You then use a CLI command to create the registry, passing the configuration file as parameter. To make the registry available to the Developer Portal, you must enable the registry in the associated catalog.
Note:

- An OIDC registry cannot be used to secure APIs on the gateway.
- Because the interaction with the third-party OIDC provider is handled by the Management server, the Management server is the application from the point of view of the third-party OIDC provider. Your OIDC redirection endpoint is used by the authorization server to send the token to the Management server, so it must be accessible to the OIDC provider through your firewall. When you register your application with the third-party OIDC provider, you are required to supply the associated OIDC redirect URI, `https://consumer.mycompany.com/consumer-api/oauth2/redirect` for example. However, this information is not available until you have created your OIDC user registry in API Connect. You must first register your application without this information and then update it later, as detailed in the instructions on this page.

Logging in to the Management server

Before you can create an OIDC user registry, you must log in to your management server from the CLI. Use the following command:

```
apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
For full details of the **apic login** command, see Logging in to a management server.

Defining your OIDC registry configuration

Define the configuration of your OIDC user registry in a YAML file. At a minimum, the YAML file must include the settings from the following list. For additional settings, see the tables provided with the UI version of the procedure in this topic.

```
title: registry_title
integration_url: oidc_integration_url
case_sensitive: case_sensitivity_setting
email_required: true_or_false
email_unique_if_exist: true_or_false
configuration:
  client_id: 'app_client_id'
  client_secret: 'my-client-secret'
  provider_type: oidc_provider_type
```

where:

- *registry_title* is your chosen descriptive title for the user registry.
- *oidc_integration_url* is the OIDC integration URL in your API Connect configuration. You can determine the OIDC integration URL by using the following CLI command:

  ```
  apic integrations:list --server mgmt_endpoint_url --subcollection user-registry
  ```

- *case_sensitivity_setting* determines whether your user registry is case-sensitive. Valid values are:
  - **true**
  - **false**

  To ensure proper handling of user name capitalization, you **must** ensure that your case-sensitivity setting here matches the setting on the backend OIDC provider:
  - Only set **case_sensitive** to **true** if the backend OIDC provider supports case-sensitivity.
  - Set **case_sensitive** to **false** if the backend OIDC provider does not support case-sensitivity.

  Note: After at least one user has been added to the registry, you cannot change this setting.
- *email_required* determines whether an email address is required as part of the user onboarding process. Valid values are:
  - **true**
  - **false**

  If set to **true**, the source identity provider must supply the email address as part of the authentication process during onboarding.
  Note: An email address is not required by default for onboarding to the Cloud Manager or the API Manager, but it is required for onboarding to the Developer Portal.
- *email_unique_if_exist* determines whether email addresses must be unique within the user registry. Valid values are:
  - **true**
  - **false**

  Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account.
- *app_client_id* is the client ID of the application that is registered with the OIDC server, and must be in string format.
- *my-client-secret* is the client secret of the application that is registered with the OIDC server, and must be in string format.
- *oidc_provider_type* is the type of OIDC provider; specify one of the following values:
  - **facebook**
  - **github**
  - **google**
  - **linkedin**
  - **slack**
  - **twitter**
  - **windows_live**
  - **standard**

  Note:
  - Use the **twitter** provider type if your OIDC provider is X (formerly known as Twitter).
  - Use the **standard** provider type for any OIDC provider that is compliant with the OIDC standard.
    If the provider type is **standard**, you must include the following additional properties in the **configuration** section of your YAML file:

    ```
    authorization_endpoint: 'oidc_auth_endpoint'
    token_endpoint:
      endpoint: 'oidc_token_endpoint'
    ```

    where:
    - *oidc_auth_endpoint* is the authorization endpoint on the OIDC server, and must be in string format.
    - *oidc_token_endpoint* is the token endpoint on the OIDC server, and must be in string format.

Default OIDC configurations

For each OIDC provider type, API Connect assumes a default configuration, but you can override the default configuration properties in your YAML file.

Note: Endpoints are subject to change; when you create a registry you must verify the default endpoint provided by API Connect.
API Connect makes the best effort to update endpoints for the known OIDC user registry types. However the OIDC provider can change the endpoint at any time. It is the customer's responsibility to confirm that the endpoints provided by API Connect are supported by the OIDC provider and to update the OIDC configuration as needed.

The default configurations are as follows:

- Facebook

  ```
  authorization_endpoint: 'https://www.facebook.com/v3.1/dialog/oauth'
  token_endpoint:
    endpoint: 'https://graph.facebook.com/v3.1/oauth/access_token'
  userinfo_endpoint:
    endpoint: 'https://graph.facebook.com/me'
  ```

```
scope: email public_profile
field_mapping:
  username: email
  email: email
  last_name: last_name
  first_name: first_name
```

- GitHub

```
authorization_endpoint: 'https://github.com/login/oauth/authorize'
token_endpoint:
  endpoint: 'https://github.com/login/oauth/access_token'
userinfo_endpoint:
  endpoint: 'https://api.github.com/user'
scope: 'read:user user:email'
field_mapping:
  username: login
  email: email
  last_name: name
  first_name: name
```

- Google

```
authorization_endpoint: 'https://accounts.google.com/o/oauth2/v2/auth'
token_endpoint:
  endpoint: 'https://www.googleapis.com/oauth2/v4/token'
scope: openid profile email
field_mapping:
  username: email
  email: email
  last_name: family_name
  first_name: given_name
```

- LinkedIn
  Some settings differ between the legacy configuration and the newer configuration. Be sure to use the correct settings for the LinkedIn protocol that you are using.

  - New OIDC protocol:

    ```
    authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
      credential_location: form_body
      field_mapping:
        email: email
        username: name
        first_name: given_name
        last_name: family_name
      scope: openid profile email
      token_endpoint:
        endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
      userinfo_endpoint:
        endpoint: 'https://api.linkedin.com/v2/me'
    ```

  - Legacy OIDC protocol:

    ```
    authorization_endpoint: 'https://www.linkedin.com/oauth/v2/authorization'
    token_endpoint:
      endpoint: 'https://www.linkedin.com/oauth/v2/accessToken'
    userinfo_endpoint:
      endpoint: 'https://api.linkedin.com/v1/people/~:(id,first-name,last-name,picture-url,public-profile-url,email-
    address)?format=json'
    scope: r_basicprofile r_emailaddress
    field_mapping:
      username: emailAddress
      email: emailAddress
      last_name: lastName
      first_name: firstName
    credential_location: form_body
    ```

- Slack

```
authorization_endpoint: 'https://slack.com/oauth/authorize'
token_endpoint:
  endpoint: 'https://slack.com/api/oauth.access'
userinfo_endpoint:
  endpoint: 'https://slack.com/api/users.identity'
scope: identity.basic identity.email
field_mapping:
  username: user.email
  email: user.email
  last_name: user.name
  first_name: user.name
```

- X (formerly known as Twitter)
  Although Twitter re-branded to X, the URL for the OIDC endpoint remains twitter.com.

```
request_endpoint: https://api.twitter.com/oauth/request_token'
authorization_endpoint: https://api.twitter.com/oauth/authenticate'
token_endpoint:
    endpoint: 'https://api.twitter.com/oauth/access_token'
userinfo_endpoint:
    endpoint: 'https://api.twitter.com/1.1/account/verify_credentials.json'
oauth_signature_method: 'HMAC-SHA1'
field_mapping:
    email: email
    first_name: name
```

```
    last_name: name
    username: screen_name
```

- Windows Live

```
authorization_endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/authorize'
token_endpoint:
  endpoint: 'https://login.microsoftonline.com/common/oauth2/v2.0/token'
scope: openid offline_access profile email
field_mapping:
  email: email
  username: preferred_username
  first_name: name
  last_name: name
```

- Standard

```
response_type: code
scope: openid
field_mapping:
  username: sub
  email: email
  last_name: family_name
  first_name: given_name
credential_location: auth_header
```

  Although this is the default configuration for the **standard** provider type, you should contact your OIDC provider to obtain the details of the fields that you need to define.

Creating your OIDC user registry

To create your OIDC user registry, use the following CLI command:

```
apic user-registries:create --server mgmt_endpoint_url --org organization_name oidc_config_file
```

where:

- *mgmt_endpoint_url* is the platform API endpoint URL.
- *organization_name* is the value of the **name** property of your provider organization.
- *oidc_config_file* is the name of the YAML file that defines the configuration of your OIDC user registry.

On completion of the registry creation, the command displays the following summary details:

```
registry_name registry_url
```

By default, the *registry_name* is derived from the **title** property in the configuration YAML file but you can override this by including a **name** property in the file. The *registry_url* is an internal URL that API Connect assigns to the registry.
After you have created your OIDC user registry, you must update your application registration with the third party OIDC provider to include the OIDC redirect URI; you can obtain this information by using the following command, which displays the details of the registry in the command window:

```
apic user-registries:get --server mgmt_endpoint_url --org organization_name registry_name --output -
```

The required **oidc_redirect_uri** value is in the **consumer:** section; for example:

```
consumer:
  oidc_redirect_uri: https://consumer.mycompany.com/consumer-api/oauth2/redirect
```

Enabling your OIDC registry in a Catalog

To make your OIDC registry available for onboarding and authenticating Developer Portal users, you must enable it in the Catalog that is associated with that Developer Portal. Complete the following steps:

1. Determine the URL of your OIDC user registry by using the following command:

   ```
   apic user-registries:list --server mgmt_endpoint_url --org organization_name
   ```

2. Enter the following command (the terminating hyphen character means that the command takes input from the command line):

   ```
   apic configured-catalog-user-registries:create --server mgmt_endpoint_url --org organization_name --catalog catalog_name -
   ```

   where *catalog_name* is the value of the `name` property of the required Catalog. The command returns

   ```
   Reading CONFIGURED_CATALOG_USER_REGISTRY_FILE arg from stdin
   ```

3. Enter the following data, followed by a new line:

   ```
   user_registry_url: oidc_registry_url
   ```

   where *oidc_registry_url* is the URL of your OIDC registry, obtained in step 1.
4. Press **CTRL D** to terminate the input.

For details of all the **apic user-registries** and **apic configured-catalog-user-registries** commands, see apic user-registries and apic configured-catalog-user-registries.

You can also complete the operations described in this topic by using the API Connect REST APIs; see the API Connect REST API documentation.

Modifying the configuration details for a user registry

You modify the configuration details for a user registry by using the User Registries page in the API Manager user interface of IBM® API Connect.

About this task

If you want to authenticate an API Connect Catalog with a user registry that is not yet defined in API Connect, you define the registry when you configure the Catalog; for details, see Working with user registries.
Note: User registries shared from the Cloud Manager cannot be edited.
To modify the configuration details for a user registry that is already defined, you use the User Registries page.

Procedure

To modify the configuration details for a user registry, complete the following steps:

1. In the API Manager, click Resources, and then select User Registries.
2. On the User Registries page, click the options icon for the user registry that you want to edit, then click Edit.
   The details of the registry are displayed.
3. Modify the configuration details as required, then click Save to save your changes.

Password lockout criteria

You can be locked out of your account if you attempt to log in and fail consecutively.

Note: Account lock out applies only for local user registries.
The length of time that you are locked out of using the account is based on the number of consecutive attempts that you fail. The length of time increases as the number of consecutive failed attempts increases.

For example, you can be locked out for 15 seconds if you have five consecutive failed attempts, or 32 minutes for 12 consecutive failed attempts.
Note: External user registries, such as LDAP, might enforce their own lockout criteria.

Configuring a native OAuth provider

Native OAuth providers are configured and managed by you within your cloud.

About this task

A native OAuth provider object provides settings for OAuth processing operations such as generating and validating OAuth tokens. An OAuth provider object is referenced by an OAuth security definition to protect an API. When a native OAuth provider is used, the OAuth operations are performed natively by API Connect.

Every OAuth provider object has a backing API. Your configuration here automatically updates the OpenAPI document of the API. You can edit the OpenAPI document directly by navigating to the Resources > OAuth Providers page, selecting your OAuth provider, then clicking API Editor.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

When a published API references an OAuth provider object, the backing API is automatically made available in the gateway.

Token requests and client_id checks:
If you are calling an OAuth endpoint, the sequence of `client_id` checks in the token request is as follows:

1. Check both body and query.
   - If found in only the body, validate and return a `200` or appropriate return code.
   - If found in only the query, return a `wrong location` error.
   - If found in both the body and the query, return a `more than one location` error.
2. When not found in the body or the query, check the `Authorization` header.
   - If found in this header, validate and return a `200` or appropriate return code.
3. Not found, return the appropriate error and code.

The best location for the `client_id` is in the request body.
One of the following roles is required to configure a native OAuth provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

Note:

- The OAuth provider logs analytics data for failure cases, but does not log successful cases. Activity log policies that call for logging of analytics data upon success do not apply for the OAuth provider.
- You must ensure the OAuth Provider is configured in the Sandbox Catalog before using the OAuth Provider in a non-Sandbox Catalog.

Procedure

1. In the API Manager, click Resources.
2. Click OAuth Providers > Add > Native OAuth provider.
   a. Complete the following parameters for the first screen, then click Next.

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
Gateway Type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see API Connect gateway types. OAuth Providers apply to one gateway type.

   b. In the next screen, enter the following additional configuration parameters, then click Next.

Field	Description
Authorize Path	`/oauth2/authorize/` is the standard OAuth endpoint to login to account
Token Path	`/oauth2/token/` is the standard OAuth endpoint to exchange code for access token.

Supported grant types	• Implicit - An access token is returned immediately without an extra authorization code exchange step.
	• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.
	• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."
	• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.
	• API Gateway only Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.
	Note: To use the Resource owner - JWT option, complete the following steps:
	  • In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT.
	  • Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type.
	  For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).

Supported client types	• Confidential - Client can maintain secure credentials on a secure server • Public - Client credentials are not secure.

   c. Enter the scopes in the next screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

   d. Enter the parameters for User Security in the next screen. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application grant type. Click Next when done.

Field	Description
Identity Extraction	Determines how the user credential is extracted: • Basic Authentication - HTTP basic authentication (requires no additional configuration) • Default HTML Form - Use default login form for user name and password • API Gateway only Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here API Connect context variables Note: DataPower API Gateway only. Not available for DataPower Gateway (v5 compatible). • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML login form for user security. • Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see Authenticating and authorizing through a redirect URL. • API Gateway only Disabled - do not collect the user credential Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint **must** maintain the order of the query parameters before the `state_nonce` query parameter, otherwise the authorization fails.
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry. For a DataPower API Gateway, you have the option to disable authentication with a user registry.
Authorization	The following methods for extracting the user credential are available: • Authenticated - Authorize authenticated users automatically. • Default HTML Form - Use default HTML form to authorize. • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML authorization form for user security. • API Gateway only Disabled - Disable authorization.

3. Review the Summary for the native OAuth provider configuration.
4. Click Back to make changes.
5. Click Finish to save the basic configurations and to proceed to the Advanced Parameters for a native OAuth Provider.
   If you want to specify the native OAuth Provider in your Catalog, see Specifying the OAuth Providers.

Results

You can use the OAuth Provider to secure the APIs in catalog.

- **Configuring basic settings for a native OAuth provider**
  You can update the identification details and basic configuration settings for a native OAuth provider.
- **Configuring scopes for a native OAuth provider**
  Access tokens contain authorization for specific scopes.
- **Configuring user security for a native OAuth provider**
  Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.
- **Configuring tokens for a native OAuth provider**
  Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.
- **Configuring token management and revocation for a native OAuth provider**
  Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.
- **Configuring introspection for a native OAuth provider**
  Define an introspection path to allow the metadata for an access token to be examined.
- **Configuring metadata for a native OAuth provider**
  Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- **Configuring the OIDC parameters for a native OAuth provider**
  Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.
- **Editing a native OAuth provider by using the API Editor**
  You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

Configuring basic settings for a native OAuth provider

You can update the identification details and basic configuration settings for a native OAuth provider.

About this task

One of the following roles is required to configure the basic settings for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the basis settings pages for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the basic settings for an existing native OAuth provider. If you want to update the basic settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. To modify the identification details, click Info in the sidebar menu, then update the following fields as required:

Field	Description
Title	Enter a title for the native OAuth provider.
Name	This field is auto-populated by the system.
Description (optional)	Enter a brief description.
Base path (optional)	The base path is the URL segment of the API that is shared by all operations in the API. It does not include the host name or any additional segments for paths or operations. The base path must be unique for a given catalog. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.

2. To modify the basic configuration settings, click Configuration in the sidebar menu, then update the following fields as required:

Field	Description
Authorize Path	`/oauth2/authorize/` is the standard OAuth endpoint to login to account
Token Path	`/oauth2/token/` is the standard OAuth endpoint to exchange code for access token.

Supported grant types	• Implicit - An access token is returned immediately without an extra authorization code exchange step.
	• Application - Application to application. Corresponds to the OAuth grant type "Client Credentials." Does not require User Security.
	• Access code - An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type "Authorization Code."
	• Resource owner - Password - The user's username and password are exchanged directly for an access token, so can only be used by first-party clients.
	• API Gateway only Resource owner - JWT - A verified signed JSON Web Token is exchanged directly for an access token.
	Note: To use the Resource owner - JWT option, complete the following steps:
	  • In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT.
	  • Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type.
	  For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).
	
	Tip: If you plan to configure OpenID Connect (OIDC) for a Native OAuth provider, include at least one of the following grant types: Implicit, Access code.

Field	Description
Supported client types	• Confidential - Client can maintain secure credentials on a secure server • Public - Client credentials are not secure.

DataPower Gateway (v5 compatible) only Note: If the gateway type is DataPower® Gateway (v5 compatible) and, when the native OAuth provider was created, only the Application grant type was selected, you cannot add further grant types until you configure the user security settings. In particular, you must specify the user registry for authenticating application users. To configure the user security settings, complete the following steps:

a. Click User Security in the sidebar menu, then click Edit.
b. Update the user security settings as required; for more details, see Configuring user security for a native OAuth provider.
c. Click Save when done.

3. Click Save when done.

Configuring scopes for a native OAuth provider

Access tokens contain authorization for specific scopes.

About this task

The client applications can request only the scopes or a subset of the scopes that you define here. The scopes are included in the access tokens that are generated from the provider. When an OAuth protected API is invoked, the gateway checks the scopes carried in the access tokens against the list of allowed scopes in the security definition for the API to determine whether to grant access.

In addition, you can enforce advanced scope checks. The advanced scope check URLs are invoked after application authentication or after user authentication based on which URLs are configured. The final scope permission that is granted by the access token is the result of all scope checks.

Per IETF RFC 6749, the value of the scope parameter is a list of space-delimited, case-sensitive strings. For more information, see The OAuth 2.0 Authorization Framework.

One of the following roles is required to configure scopes for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the scope settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the scope settings for an existing native OAuth provider. If you want to update the scope settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click Scopes in the sidebar menu. The currently configured scopes are listed. Review and update the scopes as required.

Field	Description
sample_scope_1	Scope for token
additional scopes	Scope for token

In the Default scopes section, select the default scopes to be used if the API request doesn't contain any scopes. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.
If the user authorization method is set to Default HTML Form in the User Security settings, all scopes specified here are added automatically to the authorization consent form.

2. Advanced scope check before token generation. This setting specifies the scope check endpoint where additional scope verification is performed in addition to the basic scopes. The advanced scope check URLs are invoked after application authentication or after owner authentication based on which URLs are configured. The scopes are included in the token and will overwrite any previous scopes.

Field	Description
Application scope check	Allow extra verification by running a scope check from an endpoint. Enter the endpoint and an optional TLS Profile to use for an application scope check.
Owner scope check	Further refine the scope with an additional check. Enter the endpoint and an optional TLS Profile to use for an owner scope check.

For more information about scope, see Scope

3. Advanced scope check after token generation. This setting specifies an additional scope check at the API consumer level to verify compliance with the scope requirements of the API.

Field	Description
Enabled	Select the check box to enable the advanced scope check after token validation. Enter an optional default validator endpoint.
Use endpoint from API	Select the check box to use the endpoint from the API, or clear the check box to override the endpoint from the API.

For more information about scope, see Scope

4. Click Save when done.

Results

You can use the OAuth Provider with these scopes to secure the APIs in catalog.

Configuring user security for a native OAuth provider

Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization.

About this task

User security authenticates the user. It is required for the Implicit, Access code, and Resource owner - Password grant types. It is not used for the Application or Resource owner - JWT grant types.

One of the following roles is required to configure user security for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the user security settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the user security settings for an existing native OAuth provider. If you want to update the user security settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click User Security in the sidebar menu.
2. Specify the following parameters for User Security. Define the settings to use to extract the application users' credentials, authenticate their identities, and grant authorization. User Security is not required for the Application or Resource owner - JWT grant types. Click Next when done.

Field	Description
Identity Extraction	Determines how the user credential is extracted: • Basic Authentication - HTTP basic authentication (requires no additional configuration) • Default HTML Form - Use default login form for user name and password • API Gateway only Context variable - Specify which variable contains the user name and password. API Connect OAuth context variables as listed here API Connect context variables • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. For instructions on creating a custom form, see Creating a custom HTML login form for user security. • Redirect - Enter an endpoint to redirect to a third-party identity provider. For more information, see Authenticating and authorizing through a redirect URL. • API Gateway only Disabled - do not collect the user credential Note: If you use either the Default HTML Form or Redirect identity extraction methods, the response from the redirect endpoint **must** maintain the order of the query parameters before the `state_nonce` query parameter, otherwise the authorization fails.
Authentication	Authenticate application users with a user registry. Select an LDAP or Authentication URL user registry or create the SampleAuthURL User Registry.
Authorization	Various methods may be used to authorize application users. For a DataPower® API Gateway, the following methods for extracting the user credential are available: • Authenticated - Authorize authenticated users automatically. • Default HTML Form - Use default HTML form to authorize. If you select the Default HTML Form method, all scopes that are specified in the Scopes settings are added automatically to the authorization consent form. • Custom HTML Form - Enter the endpoint and select an optional TLS profile for a custom HTML form. • API Gateway only Disabled - Disable authorization.

3. Click Save when done.

Results

You can use the OAuth Provider to secure the APIs in catalog.

Configuring tokens for a native OAuth provider

Set time to live for access tokens and refresh tokens, and a time period for maximum consent for all tokens.

About this task

Access tokens are granted to the client application to allow the application to access resources on behalf of the application user. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or more narrow scope. You can also specify how long the consent given by the combination of any number of access and refresh token remains valid.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the token settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the token settings for an existing native OAuth provider. If you want to update the token settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click Tokens in the sidebar menu.
2. Define the settings to configure tokens.

Field	Description
Access tokens time to live	Enter the expiration time period in seconds for access tokens.
API Gateway only One time use access token	Click the check box to enable one time use for the access token. Access tokens are multiple use by default which allows them to be used for multiple requests. When one time use is enabled, the access token will be consumed after one use. The OAuth flow will need to be repeated to obtain another access token. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: • Configuring token management and revocation for a native OAuth provider (Cloud Manager) • Configuring token management and revocation for a native OAuth provider (API Manager)
Refresh tokens	Click the check box to enable Refresh tokens. Set the Count to limit the number of times a refresh token can be issued. Set the Refresh Token Time to Live value to determine the time to live, or expiration time period, for each refresh token in seconds.
One time use refresh token	Clear the check box to disable one time use for the refresh tokens. Refresh tokens are one time use by default which allows them to be used one time only to generate an access token and a new refresh token. When refresh token one time use is disabled then the refresh token count is limited to one and the refresh token can be used multiple times to generate new access tokens, however, another refresh token will not be generated unless the initial OAuth flow (Authorization Code or Password) is repeated. Note: If you select this option, you must also enable token management; see one of the following topics, depending on the user interface you are using: • Configuring token management and revocation for a native OAuth provider (Cloud Manager) • Configuring token management and revocation for a native OAuth provider (API Manager)
Maximum consent	Click the check box to enable Maximum consent and enter the Maximum Consent Time to Live value in seconds. This is the time to live, or expiration time period, for all tokens, both access and refresh.
API Gateway only Token secret	Click the check box to select the Shared Secret which was configured for the gateway. If no Shared Secret was entered in the Gateway Configuration, then enter an key name and key value to use as the token secret.
API Gateway only Proof Key for Code Exchange	Proof Key for Code Exchange (PKCE) is a method to protect OAuth 2.0 public clients from an authorization code interception attack when they use Authorization Code grant requests. You can enable this extension when deploying with the DataPower® API Gateway. For more information, see RFC 7636. Select the options for your OAuth Providers: • Enable proof key for code exchange If selected, enforces PKCE when submitted in Authorization Code grant requests. • Always required If selected, requires PKCE in all Authorization Code grant requests. • Allow plain Select this check box to allow the plain challenge method in Authorization Code grant requests.

3. Click Save when done.

Results

You can use the OAuth Provider to secure the APIs in a catalog.

Configuring token management and revocation for a native OAuth provider

Select whether to use a native gateway (DataPower) or third party endpoint for token revocation.

About this task

Token management enables you to prevent replay attacks by configuring token revocation. API Connect supports token revocation using a native gateway (DataPower) or a third party endpoint. For a native gateway, quota enforcement is used to manage tokens. For a third party endpoint, a URL to an external service is used to manage tokens.

For more information, see the IETF RFC 7009 OAuth 2.0 Token Revocation.

Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

One of the following roles is required to configure token management and revocation for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the token management settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the token management settings for an existing native OAuth provider. If you want to update the token management settings for an

existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Select Token Management in the sidebar menu.
2. Enable token management by selecting the check box.
3. From the Type list, select either Native or External. Native points to DataPower as the token storage location; External points to a revocation URL for token storage.
   Note: If you are using the API Manager user interface then, for the External option to be available, you must be using DataPower® API Gateway Version 10.0.1.0 or later, and the gateway service must be enabled in the Sandbox Catalog; for details on how to enable a gateway service in a Catalog, see Creating and configuring Catalogs
4. For Native, select one or both of the Resource owner revocation path and Client revocation path.
   - Resource owner revocation path - Uses the standard OAuth revocation path to allow the resource owner (end user) to revoke the application permission.
   - Client revocation path - Uses the standard OAuth revocation path to allow the client (application) to revoke a single token when the application closes.

   For more information about managing tokens with the Native DataPower Gateway, see Token management with the native DataPower Gateway.
5. For External, the settings depend on the gateway type, as follows:
   DataPower API Gateway:
   - Endpoint - the URL of the external management endpoint.
   - TLS Client Profile (optional) - the TLS client profile to secure connections.
   - Security - how to secure connections. The only supported method is basic authentication.
   - Basic authentication username (optional) - the user name for authentication.
   - Basic authentication password (optional) - the password for authentication.
   - Basic authentication request header name (optional) - the request header that contains the authentication string; if you supply both a request header name and user name/password, the request header authentication method is used.
   - Custom header pattern (optional) - the name pattern of the headers to use for sending additional information to the external management service.
   - Cache type - the cache type to control whether and how to cache positive responses. If you select Time to live, specify how long to keep responses in the cache; the default value is 900 seconds.
   - Fail on error - if selected, processing is stopped if the connection to the external management service fails.

   Note: For details of the JSON format that is required when exchanging messages with the external management service, see JSON format to exchange messages with the external management service.
   DataPower Gateway (v5 compatible):
   - Endpoint - Enter the URL to an external web server that contains information about access or refresh tokens. API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing. For more information see Token revocation.
   - TLS Client Profile - Select a TLS profile to verify the external endpoint.
6. Click Save when done.

Results

You can use the OAuth Provider to secure the APIs in a catalog.

Configuring introspection for a native OAuth provider

Define an introspection path to allow the metadata for an access token to be examined.

About this task

Token introspection allows an authorized holder of an access token to examine the contents of tokens using an introspection path. The access token to introspect must be obtained through the native OAuth provider. Introspection provides context for the token by allowing an authorized protected resource to query the authorization server to determine the set of metadata for a given token. The metadata includes whether or not the token is currently active, the scopes assigned to the token, and the authorization context in which the token was granted (including who authorized the token and which client it was issued to). API Connect token introspection conforms to IETF RFC 7662. See OAuth 2.0 Token Introspection.

One of the following roles is required to enable introspection for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the introspection settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the introspection settings for an existing native OAuth provider. If you want to update the introspection settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click Introspection in the sidebar menu.
2. Select the check box to enable Introspection. The OAuth standard path for introspection, /oauth2/introspect is automatically entered. This path will be used when another entity inspects the token contents.
3. Click Save when done.

Results

Tokens will be queried using the /oauth2/introspect path. You can use the OAuth Provider to secure the APIs in a catalog.

Configuring metadata for a native OAuth provider

Use Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

About this task

Configure an Authentication URL or an External URL from which custom metadata is collected for inclusion in the token. The metadata is either stored inside the access token or it is sent along with the access token to the client application. For more information about how the metadata is collected, see OAuth external URL and authentication URL.

Following are examples of metadata that can be included with the access token:

- Metadata about the authenticated resource owner
- Grant type that was used to obtain the token
- A confirmation code to be provided to the client application

One of the following roles is required to configure metadata collection for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can select the metadata settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the metadata settings for an existing native OAuth provider. If you want to update the metadata settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click Metadata in the sidebar menu.
2. Select Collect metadata to enable metadata collection.
3. The Authentication URL user registry is selected by default and is required. For more information about the Authentication URL, see Authentication URL user registry.
4. Select the External URL to collect metadata from an external URL. Enter the endpoint and an option TLS Client Profile.
5. API Gateway only If required, override the default Header name token value. The value of this header, if returned in the response from the OAuth endpoint, is placed in the response payload and indicated as `metadata`.
6. API Gateway only If required, override the default Header name payload value. The value of this header, if returned in the response from the OAuth endpoint, is placed within the access token and indicated as `miscinfo`.
7. Save the OAuth Provider.
8. Click Save when done.

Results

You can use the OAuth Provider to secure the APIs in a catalog.

Configuring the OIDC parameters for a native OAuth provider

Open ID Connect (OIDC) provides an additional authentication protocol based on OAuth 2.0. OIDC provides user information encoded in a JSON Web Token, or JWT.

About this task

When you enable OpenID connect, a template is provided for generating ID tokens along with access tokens and the required assembly policies are automatically created. You can customize the policies to suit your needs in the API Editor. The sample key provided is for test purposes only and is used to sign the JWT token.

One of the following roles is required to configure an OIDC template for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the System > Manage permissions

Note: You can configure OIDC parameters only if the selected grant types for the native OAuth provider include at least one of the Implicit or Access code grant types; see Configuring basic settings for a native OAuth provider.
You can select the OIDC settings page for a native OAuth provider immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the OIDC settings for an existing native OAuth provider. If you want to update the OIDC settings for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click OpenID Connect in the sidebar menu.
2. Select the initial check box to configure an OIDC Template. Enter the following parameters:

Field	Description
API Gateway only Support hybrid response types (optional)	Select the response types for the OpenID Connect hybrid flow to be supported by this OAuth provider.
API Gateway only Auto Generate OIDC API Assembly	Select this option to generate the full OIDC assembly. Leave this option unselected to simply enable OIDC support in the OAuth provider and allow the developer to implement their own assembly.
DataPower Gateway (v5 compatible) only ID token issuer	Descriptive text to indicate the source of the key.
DataPower Gateway (v5 compatible) only ID token signing key	Specify the JSON Web Key (JWK) to be used to sign the ID token.
DataPower Gateway (v5 compatible) only ID token signing algorithm	Select the algorithm used to sign the token.

3. Click Save when done. You can edit the policies by using the API Editor.

Results

You can use the OAuth Provider to secure the APIs in a catalog.

Editing a native OAuth provider by using the API Editor

You can edit the source and assembly policies for the Native OAuth Provider using the API editor.

About this task

If you have configured an OIDC template, you can customize it in API Editor. In the API Editor, the Source tab allows you to edit the code for the configuration using a text editor. The API Assemble tab provides a graphical drag-and-drop editor (identical to the one in API Manager) that allows you to add additional elements to the assembly for the OAuth Provider.

Note: Take care when modifying the code directly on the Source tab of the API Editor because validation is limited. For example:

- If you change the name of auto generated assembly actions in the source code, the assembly will be prevented from updating dynamically when the OAuth provider settings are modified.
- You must ensure that the OAuth provider name matches the value specified in the `oauth-provider-settings-ref` field in each OAuth assembly action.

One of the following roles is required to configure tokens for a native OAuth Provider:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

You can modify the native OAuth provider configuration by selecting the API Editor page immediately on completion of the creation operation detailed in Configuring a native OAuth provider, or you can update the configuration for an existing native OAuth provider. If you want to update the configuration for an existing native OAuth provider, complete the following steps before following the procedure described in this topic:

1. Click Resources > OAuth Providers.
2. Select the required native OAuth provider.

Procedure

1. Click API Editor in the sidebar menu.
2. In the Source tab, view and edit the policies to customize the behavior for the OAuth provider.
3. In the API Assemble tab, use the drag and drop editor to add additional policies to the OIDC behavior.
   Note: If you add a policy that references a TLS profile, an **`invoke`** policy for example, then when you publish an API that uses this OAuth provider, you must ensure that the TLS profile is enabled for the Catalog to which you publish the API. For details on how to enable a TLS profile in a Catalog, see Creating and configuring Catalogs.
4. Save the edits.
5. Click Save when done.

Results

You can use the OAuth Provider to secure the APIs in a catalog.

Configuring a third-party OAuth provider

Enter the secure endpoints to provide OAuth authentication from a third party.

About this task

One of the following roles is required to configure OAuth Providers:

- Organization Administrator
- Owner
- Custom role with the Settings > Manage permissions

Procedure

1. In the API Manager, click Resources.
2. Select OAuth Providers > Add > Third party OAuth Provider.
   a. Complete the following parameters for the first screen and click Next.

Field	Description
Title	Enter a descriptive title for the gateway service. This title will be displayed on the screen.
Name	This field is auto-populated by the system and used as the internal field name.
Supported grant types	Select from the following options: • Implicit: An access token is returned immediately without an extra authorization code exchange step. • Application: Application to application. Corresponds to the OAuth grant type Client Credentials. Does not require User Security. • Access code: An authorization code is extracted from a URL and exchanged for an access code. Corresponds to the OAuth grant type Authorization Code. • Resource owner - Password: The user's username and password are exchanged directly for an access token, so can only be used by first-party clients. • API Gateway only Resource owner - JWT: a JSON Web Token (JWT) Bearer Token is used as a means for requesting an OAuth 2.0 access token, and for client authentication, as defined by the JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. Note: To use the Resource owner - JWT option, complete the following steps: • In the Supported grant types field, select both Resource owner - Password and Resource owner - JWT. • Edit the API definition and add a security scheme that specifies oauth2 as the security definition type and select Resource owner - Password as the flow type. For instructions on defining an OAuth2 security scheme for an API, see Defining OAuth2 security scheme components (OpenAPI 3) or Defining OAuth2 security schemes (OpenAPI 2).
Gateway type	Select the gateway type, either DataPower® Gateway (v5 compatible) or DataPower API Gateway. For information about types of gateways, see API Connect gateway types. OAuth Providers apply to one gateway type.

   b. Specify configuration settings for the endpoints, and then click Next.

Field	Description
Authorization URL	An authorization URL where the resource owner grants authorization to the client application to access a protected resource. Example: `https://example.com/oauth2/authorize`
Token URL	A token request URL where the client application exchanges an authorization grant for an access token. Example: `https://example.com/oauth2/token`
Introspect URL	The introspection URL is where the API gateway validates the access tokens that are issued by the third party provider. Example: `https://example.com/oauth2/introspect` For more information on integrating third party OAuth providers for introspection, see OAuth introspection for third-party OAuth providers.
API Gateway only Introspect cache type	The cache type determines how long responses from the third party provider are cached, if at all. Select one of the following options: • No cache (default): Responses are not cached. • Protocol: Defined by the `cache-control` header in the provider response. • Time to live: Defined by the provider.
API Gateway only Cache Time to Live	The length of time, in seconds, for which provider responses are cached, if the Introspect cache type is set to Time to live. The default value is 900.
TLS Profile (optional)	Select an optional TLS profile for communicating with the third party provider.
Security	Default is Basic Authentication.
Basic authentication request header name	The `x-introspect-basic-authorization-header` is available to provide a user-configured HTTP Basic authorization header.
API Gateway only Basic authentication username (optional)	The default user name for HTTP Basic authentication.
API Gateway only Basic authentication password (optional)	The default password for HTTP Basic authentication.
API Gateway only Token validation	Specifies the method used to determine the success of the introspection request that is sent to the third party service to validate the provided token. Select one of the following options: • Connected: The query is successful if the status return code is `200`. • Active (default): The query is successful if the status return code is `200` and the response JSON body includes the property `active: true`.

Field	Description
API Gateway only Custom header pattern (optional)	A regular expression for request headers that are to be passed to the third-party provider; for example, `x-Introspect-*`.
API Gateway only Authorization header pass through	Select this check box if you want to retain the **`Authorization`** header for a bearer token. The default behavior is to remove this header.

c. Enter the scopes in the third screen. A scope becomes an option in the request and response for an access token. Click Add to add additional fields for scopes. Click Next when done.

Field	Description
sample_scope_1	Scope for token
sample_scope_2	Scope for token
additional scopes	Scope for token

d. Review the settings on the Summary panel.

3. Click Save and Edit to complete the configuration.

OAuth concepts for API Connect

OAuth is a token-based authorization protocol that allows third-party websites or applications to access user data without requiring the user to share personal information.

Note: In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

- **OAuth user scenario**
  Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.
- **OAuth introspection for third-party OAuth providers**
  OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM API Connect can use this feature along with the mentioned provider to protect access to the API.
- **OAuth external URL and authentication URL**
  You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.
- **Scope**
  Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.
- **Tokens**
  Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.
- **Authentication URL user registry**
  You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.
- **Custom forms for user security**
  You can create custom forms for the authorization and identity extraction phase of OAuth.
- **Securing an API with a JSON Web Token**
  There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM API Connect.
- **Troubleshooting OAuth**
  You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

Related information

- The OAuth 2.0 Framework

OAuth user scenario

Potential users of OAuth with IBM® API Connect have a number of methods to secure their API. The following scenario provides an overview of the available options.

Scenario overview

In this scenario, Alice is a user of an application. Alice can grant permission for an application to access specific information about Alice in a third-party system that is using OAuth. Depending on the type of OAuth that is supported by the target service, Alice does not enter her user name and password into the application. Instead, the application receives an access token that represents her credentials (user name and password). The application can now access the information about Alice in the target system.

For example, Alice maintains a list of books that she reads on a service that is provided by mybooks.com. Following the purchase of a smartphone, Alice installs an application on her new phone to display the book details. The phone application wants to call an API provided by mybooks.com, which can access the information. The mybooks.com API is secured by using the OAuth protocol.

To access the book details, the application must complete a two-step process:

1. The application must first obtain permission from Alice.
2. The application then uses that permission to call the target service and obtain the list of books.

In the first step, the application typically directs Alice to the provider of the target service, mybooks.com. Alice provides her user name and password, and gives permission for the application to access her information. It is important that Alice trusts that she is providing her credentials to the provider of the target service and not to an untrusted proxy application. For example, by checking that the security certificate of the website where Alice enters her credentials matches what Alice expects from the provider of the target service.

The result of this step is the access token that the application can use to call the API. The application then generates the appropriately formatted OAuth request. For example, the Authorization header, or HTTP query parameters, which includes the access token, consumer key, and signature method that are required by OAuth. This OAuth request is used to invoke the API proxy operation.

Scenario within API Connect

No changes to the definition of your API operation are required to support this scenario.

1. Alice grants permission for the application to access her information *before* the invocation of the API.
2. When the application provides the Authorization header, or query parameters, containing the OAuth details about the call to the operation endpoint, the header is automatically passed through to the target service without any additional configuration.

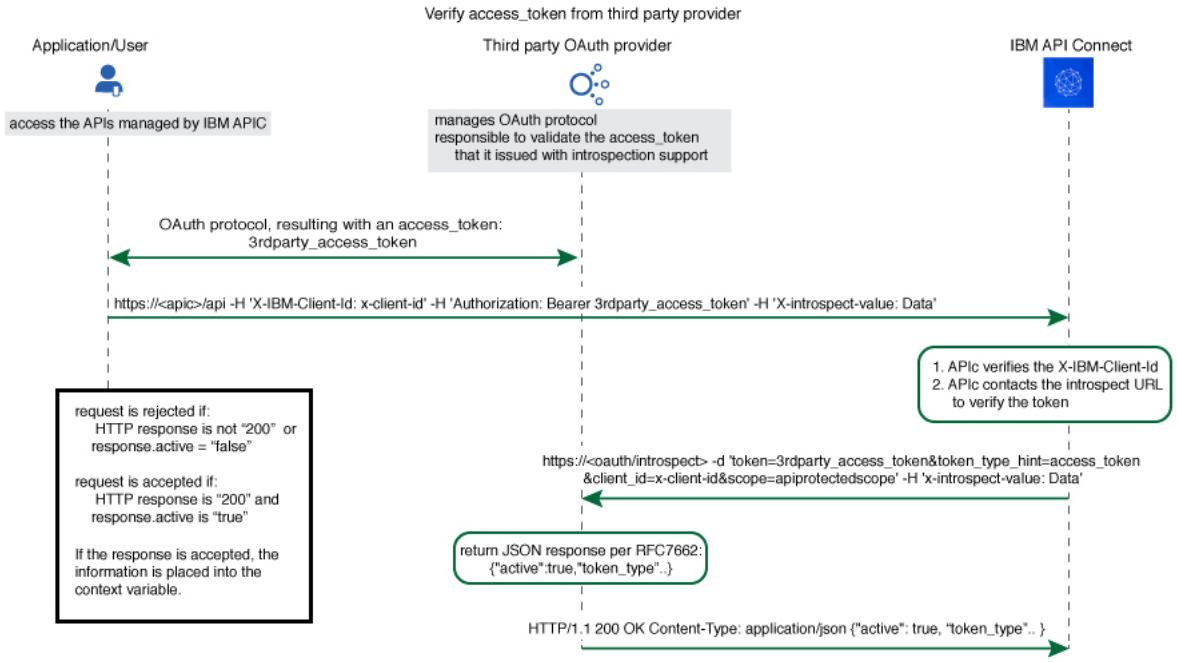
OAuth introspection for third-party OAuth providers

OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. IBM® API Connect can use this feature along with the mentioned provider to protect access to the API.

You can use API Connect to protect an API that is secured by using the third-party OAuth access token in accordance with Introspection specification as defined in RFC 7662. In addition to the specification, the **`x-Introspect-`** header is provided to pass other content to the third party as you require.

Authentication information can be carried in the request by configuring a basic authentication request header.

The following sequence diagram depicts the overall flow of request and response. The purpose of this diagram is only to provide a general visualization of the nature of the flow; for the precise details, refer to the explanatory information after the diagram.



The Introspect URL is configured in the Third Party OAuth provider configuration. See Configuring a third-party OAuth provider.

When an API is protected by a third-party OAuth provider, API Connect will extract the bearer token and issue an HTTP POST request to the endpoint specified in the Introspect URL field.

The GET request is protected by API Connect with this feature.

You can use a header prefix to pass information to the third-party provider. The header prefix can include a regular expression and specifies the name pattern of the headers to use for sending additional information, such as **`x-introspect-*`**.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
Authorization: Bearer tGzv3JOkF0XG5Qx2TlKWIA
x-IBM-Client-Id: xxx-xxx
```

However, if you want to use a different header prefix, specify the required value in the Custom header pattern field in the third party OAuth provider configuration. For third-party OAuth provider configuration details, see Configuring a third-party OAuth provider. The Custom header pattern feature is available only with the DataPower®

API Gateway, if you are using the DataPower Gateway (v5 compatible) the header prefix must be **x-Introspect-**.

API Connect will issue this POST request to the introspection endpoint specified in **x-tokenIntrospect**, as illustrated in the code sample:

```
POST /oauth/introspectURL HTTP/1.1
Host: apiconnect.ibm.com
Content-Type: application/x-www-form-urlencoded
x-Introspect-type: dog
x-Introspect-name: simon
x-custom-apic: petstore123
token_type_hint=access_token&token=tGzv3JOkF0XG5Qx2TlKWIA
```

The third party OAuth/OIDC provider will respond with `HTTP 200` indicating the request was successful and that payload contains the token information. API Connect honors the active claim as defined in the RFC specification.

If the value of the active claim is `true`, the token is treated as valid.

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
 Cache-Control: no-store
 Pragma: no-cache

 { "active":true,
    "token_type":"bearer",
    "client_id":"xxx-xxx",
    "username":"John Smith",
    ...
 }
```

If the value of the active claim is `false`, the token is treated as invalid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "active":false
}
```

A response code other than **HTTP 200** indicates failure to process the request.

When the OAuth token is valid and active, context variables are populated with information from the introspect JSON response. For more information, see API Connect context variables.

When contacting the introspection endpoint, API Connect uses **client_id/appId** and **client_secret/appSecret** to construct the HTTP Basic authorization header.

By default, if **x-introspect-basic-authorization-header** exists in the request, the value is used for the HTTP Basic authentication header when the introspection endpoint is contacted. API Connect verifies that the HTTP Basic authentication header value is Base64 encoded before it is sent, and encodes it if necessary as shown in the following example. If the header is already encoded, it is sent without modification.

```
GET /petstore/pet/123 HTTP/1.1
Host: apiconnect.com
x-introspect-basic-authorization-header: user:password
```

API Connect issues the following:

```
POST ..
Authorization: Basic M3JkLXBhcnR5LWNsaWVudF9pZDozcmQtcGFydHlfY2xpZW50X3NlY3JldA==token_type_hint=access_token&token= ..
```

If you are using the DataPower API Gateway, you can specify your own value for the HTTP Basic authentication header by providing the required value in the Basic authentication request header name field in the third party OAuth provider configuration. The following rules determine what authentication data is passed to the introspection endpoint:

- If there is a basic authentication header in the request, the specified credentials are used. The value must be either a string in the format ***user*:*password***, or the Base64 encoded equivalent; API Connect will Base64 encode the value if necessary before sending the request to the introspection endpoint.
- If there is no basic authentication header in the request but there are values specified in the Basic authentication username and Basic authentication password fields in the third party OAuth provider configuration, those values are used.
- If there is no basic authentication header in the request, and user name and password details are not supplied in the third party OAuth provider configuration, but **client_id** and **client_secret** values are supplied in the body of the request, these are used.
- Otherwise, an error is returned.

For third-party OAuth provider configuration details, see Configuring a third-party OAuth provider.
If either, or both, of scope and **scope validate url** are configured, and if the response is an active token with a scope claim from the third-party OAuth Provider introspection endpoint, API Connect will further enforce the scope validation in the following order:

1. If **scope** is configured for the OAuth API protection, verify the third-party scope against the scope that is configured.
2. If **scope validation url** is configured, verify the third-party scope against the scope validation URL.
3. If the third-party OAuth provider introspection endpoint does not return a scope, the Gateway will not check the scope defined in the security definition of the API where the third-party OAuth provider resource is used. For details on configuring an OAuth security definition in an API, see Creating an OAuth security definition.

For more information, see Scope.
DataPower Gateway (v5 compatible) only By default the API Connect client ID and scope are sent to the third party OAuth provider. You can suppress this behavior in either of the following ways:

- Supply a **suppress-parameters** header as follows:

  suppress-parameters: client_id

  suppress-parameters: scope

or

`suppress-parameters: client_id scope`

depending on which parameters you want to suppress.

- Define an API property called `suppress-parameters` in the API definition itself, with one of the following string values:

  `client_id`

  `scope`

  or

  `client_id scope`

  depending on which parameters you want to suppress. For information on how to define API properties, see Setting API properties.

OAuth external URL and authentication URL

You can use the Authentication URL or External URL parameters to request user-defined content from a remote server and include it in the access token or in the response payload that contains the access token.

Including custom metadata in a token

In many scenarios, custom metadata needs to be included during the access token generation process. The metadata is either stored inside the access token or it is sent along with the access token to the client application. The client application can then send that access token, or the metadata in the payload, in a subsequent request to IBM® API Connect where the metadata is retrieved, validated, or sent to the downstream systems as required.

Examples include, but are not limited to:

- When resource owners are authenticated, metadata about the authenticated resource owner needs to be stored within the access token.
- The grant type that was used to obtain the token is another example of metadata stored within the access token.
- A confirmation code that needs to be sent to the client application along with access token is stored as metadata within the access token.

Configuring External URL or Authentication URL in API Connect to obtain metadata

Metadata can be set by using either or both of the following URLs:

- External URL - When you call the External URL, an HTTP GET request is sent and API Connect expects an HTTP 200 OK along with an optional set of the specified response headers.
- Authentication URL - When you call the Authentication URL, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.
  See: Authentication URL.

The External URL endpoint is entered in the Metadata section when you configure an OAuth Provider in the Cloud Manager or API Manager UI.

Use the following HTTP headers in the response, depending on the type of gateway you are using:

- DataPower® API Gateway:

  `X-API-OAUTH-METADATA-FOR-PAYLOAD`

  `X-API-OAUTH-METADATA-FOR-ACCESSTOKEN`

  Note: These are the default header names for the DataPower API Gateway but you can override them; see Configuring metadata for a native OAuth provider.
- DataPower Gateway (v5 compatible):

  `API-OAUTH-METADATA-FOR-PAYLOAD`

  `API-OAUTH-METADATA-FOR-ACCESSTOKEN`

The response header value from `X-API-OAUTH-METADATA-FOR-PAYLOAD` or `API-OAUTH-METADATA-FOR-PAYLOAD` is placed in the response payload and indicated as `metadata`.

The response header value from `X-API-OAUTH-METADATA-FOR-ACCESSTOKEN` or `API-OAUTH-METADATA-FOR-ACCESSTOKEN` is placed within the access token and indicated as `miscinfo`.

The two metadata response headers are case insensitive and you must escape any special characters in the string value content.

An example response payload that contains metadata along with the access token:

```
{
"token_type":  "bearer",
"access_token":  "AAEkNzhjDHYyyYy...cL0Mv6ctl37w7ZU",
"metadata":  "m:metadata-for-payload_content"
"expires_in":  3600,
"scope":  "read",
"refresh_token":  "AAEnj5SynCMybF...oEZ6JjxYax_HdNg",
}
```

This example output from the token introspection endpoint shows the contents of the access token with `"miscinfo"` containing the metadata information.

```
{
"active":  true,
```

```
"token_type":  "bearer",
"client_id":  "78c2f10f-799a-4e1f-8e0a-098634997a35",
"username":  "Fred Smith",
"sub":  "fred",
"exp":  1479850049,
"expstr":  "2016-11-22T21:27:29Z",
"iat":  1479846449,
"nbf":  1479846449,
"nbfstr":  2016-11-22T20:27:29Z",
"scope":  "read",
"miscinfo":  "m:metadata-for-accesstoken_content",
"client_name":  "MobileApp"
}
```

Input to the External URL

The following request headers are sent to the External URL.
Note: Any existing metadata values that were previously sent from the Authentication URL are also sent in the two request headers **x-existing-metadata-for-payload** and **x-existing-metadata-for-access-token**. The Metadata URL can make use of this information to create a new set of metadata values.
The two request headers that are sent to the Metadata URL are displayed in bold text.

```
X-existing-metadata-for-payload     payload-from-auth-url
X-existing-metadata-for-access-token     token-from-auth-url
X-URI-in     /org/env/miscinfo/oauth2/token (the URL that was sent to APIConnect for this particular token request)
X-METHOD-in     POST
X-POST-Body-in     client_id=client_id&grant_type=password&scope=read&username=name&password=password
X-X-Client-IP     IP_address
X-X-Global-Transaction-ID     ID_number
...
```

Note: If you are using the DataPower API Gateway rather than the DataPower Gateway (v5 compatible), the **X-POST-Body-in** header is not supported.

Retrieving Metadata in API Connect

As described in the previous example scenarios, the metadata can be retrieved from the access token in the Application API and sent to the downstream systems. Retrieval can be done in the API assembly, which is secured to accept tokens in the security definitions.

In the resource API that accepts an access token, the **miscinfo** field can be accessed in the Assembly with the **oauth.miscinfo** context variable, as in the example.

```
apim.setvariable('message.body',apim.getvariable('oauth.miscinfo'));
```

You can also use token introspection to look at the contents of the access token.

Refresh tokens and metadata

The Authentication URL (if configured for authentication) is invoked first, during authentication of the resource owner. The External URL is invoked as the last step, just before the generation of the access token. The only exception is when access tokens are generated from a refresh token. In cases where refresh tokens are used to generate new access tokens, the External URL is not invoked. The metadata from the refresh token is retained and then copied into the newly generated access token.

Identifying the source of the metadata

The metadata is prefixed with keywords to indicate whether it originated from the External URL or the Authentication URL.

- Metadata from the External URL is prefixed with **m:**
- Metadata from the Authentication URL is prefixed with **a:**

Note: When revocation is enabled, some internal details are also stored in the **miscinfo** field, in brackets within the access token as shown in the following example:

```
"miscinfo": "[tlsprofile@https://api-revoke-url:443/server/revocation-url]m:metadata-for-accesstoken_content"
```

Maximum size of the metadata

Metadata for the access token cannot exceed 512 bytes.

Metadata for the payload does not have a specific size restriction, except for when you use the Authorization code grant type. These restrictions are described in following sections.

Characters are not allowed in metadata in certain scenarios

When you use the Authorization code grant type, or when a consent form is used for implicit grant type, there is a temporary state or code where the metadata from the authentication URL is stored. API Connect internally uses two prefixes - **!ma** and **!mp** to differentiate between payload and token metadata received from the Authentication URL and store them internally in the temporary state/code. Hence these specific character sequences - **!ma** and **!mp** should not be used as the metadata itself.

Grant types and metadata

The OAuth grant types described in the following sections include **authorization code (access code)**, **implicit**, and **client credentials (application)**.

Authorization code grant type

- When metadata is included from an Authentication URL for an Authorization code grant type, as it is a three legged flow, both the content and the payload are stored within the **dp-state** and carried on to the authorization code and to the access token. Note that around 10 characters are used internally to differentiate between the metadata for payload and metadata for the access token when stored in the **dp-state**. In addition, if revocation is enabled, that will also be part of the token metadata. Hence the combined size of the token metadata, the payload metadata (including the 10 characters of internal data), and internal revocation details, cannot exceed 512 bytes in total.

  If the overall size of the metadata exceeds 512 bytes, then the access token generation succeeds, but the metadata fields contain an error message of "`metadata too large`" as shown in the example.

  ```
  "metadata":"m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-payload]"
  "miscinfo":"[r:gateway]m:error: metadata too large for AZ code grant type[Authorization Code-metadata-url-token]"
  ```

  This size restriction can be overcome when the metadata is sent from the Metadata URL and not from the Authentication URL, because the metadata is not stored in **dp-state** or in the authorization code.

  Example of the **authorization code (access code)** grant type:

  ```
   $ curl -k -d
  "grant_type=authorization_code&code=$mycode&client_id=$myid&scope=scope_introspect&redirect_uri="
  https://9.70.153.90/fei/sb/introspectp1/oauth2/token
  { "token_type":"bearer",
  "access_token":"AAEkOThlZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ20ZN5Tl_TqYFeIfB7BFf6HFgibEoOjWXXEA84oFsWuE4NY-
  RRZVdnGSaXAIYJz7s5vczfk5EV-BIb_6P_1YKm3ahrfhR5kI3sPO0uADEoseIP5-O9anUpEM5yhsayXvZbJ_6VDYz-hyXSJHTNqVj-
  PHBialoRkBD5qca6kO0fV2M", "metadata":"a:[Authorization Code-Test-auth-url-payload]", "expires_in":3600,
  "scope":"scope_introspect", "refresh_token":"AAFAg1EVMbwicr_L0fTZ4q6HZ-
  RcQygniXFC9zbSKO4wd3hcniC4KQX21X0fL2c8cnmzCZgws8xxLzNyfjQhUJNGl5C1GbIe3dwhXJdiWA0Go-
  dduhVtCbG26sJRRXyYrMeRxWnJsylBETPI8HQEN4a_D7fmxKcTVRZBvq86byg95qe1ZKyERi0Lhxdd_O4Nvss" }


  $ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
  "token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
  { "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746", "username":"anyuser",
  "sub":"anyuser", "exp":1484766368, "expstr":"2017-01-18T19:06:08Z", "iat":1484762768, "nbf":1484762768,
  "nbfstr":"2017-01-18T18:06:08Z", "scope":"scope_introspect", "miscinfo":"[r:gateway]a:[Authorization Code-Test-auth-
  url-token]", "client_name":"oauth_app" }
  ```

Implicit grant type

- When implicit grant type is used, the access token and the metadata are returned in the **location** header as a fragment, as you see in the example.

  ```
  < Location:
   https://localhost#access_token=AAEkOThlZDhhNjYtYTQ1ZS00YTMzLWE0N2QtZmE2OGZkMzQ0NzQ2buS2KfWdq-
  nYBJSi4nmPxQBtLae17tKBPRMzwP5BC386nlxpoOTE1G748ZVH6Mq_TJL3GeV3PtXTVIkWOLBJi_7tljiQfpnVfrNkovvZkhUexYmFkcDsmLSdaWxZ6Pc
  IMPAC4ojT8qV1sYV-
  ChTk36yqOx_NiKimZaDikDk7WTg&expires_in=3600&scope=scope_introspect&token_type=bearer&metadata=a%3A[Implicit-Test-
  auth-url-payload]

  $ curl -k -H "X-IBM-Client-Id: $myid" -H "X-IBM-Client-Secret: $mysecret" -d
  "token_type_hint=access_token&token=$mytoken" https://9.70.153.90/fei/sb/introspectp1/oauth2/introspect
              { "active":true, "token_type":"bearer", "client_id":"98ed8a66-a45e-4a33-a47d-fa68fd344746",
  "username":"anyuser", "sub":"anyuser", "exp":1484768365, "expstr":"2017-01-18T19:39:25Z", "iat":1484764765,
  "nbf":1484764765, "nbfstr":"2017-01-18T18:39:25Z", "scope":"scope_introspect", "miscinfo":"[r:gateway]a:[Implicit-
  Test-auth-url-token]", "client_name":"oauth_app" }
  ```

Client credentials grant type

Authentication URL will not be invoked when using client credentials grant type, as there is no resource owner. The metadata from Authentication URL is not available for this grant type. However, content returned from Metadata URL will be included as metadata.

Behavior when retrieving metadata with both an External URL and an Authentication URL

If an External URL is configured and a connection to the external server is successful, the response headers overwrite any existing metadata obtained from the Authentication URL to become the final value. Therefore, you must carefully examine the incoming request headers and create appropriate response headers from the External URL.

If an External URL is configured, but the connection to the External URL fails, then a failure message of "`error on metadata url`" is written for metadata in both the payload and the access token.

If an External URL is configured and the connection is successful, but the remote server does not send any of the specified HTTP response headers, a blank value is written for metadata in both the payload and the access token.
Attention: An External URL overwrites existing values from Authentication URL. This includes blank values.
If no External URL is configured, the metadata that is obtained from the Authorization URL is retained as the final value.

Scope

Per IETF RFC 6749, the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings.

In IBM® API Connect, scopes have no inherent meaning. Instead, scopes are defined in the OAuth Provider so that an application can request an access token that is valid for one or more of the scopes. In the secured API, scopes are listed as requirements for an access token to be considered valid. All scopes that are listed by the security definition for the API must be granted by the access token.
Note:

- An OAuth Provider must have at least one scope.
- An OAuth token will not be granted unless a default scope is configured at the provider, or one or more predefined scopes are included in the grant request.
- If no default scope is defined, and the request does not contain a scope, an invalid scope error is returned for the request.

OAuth provider

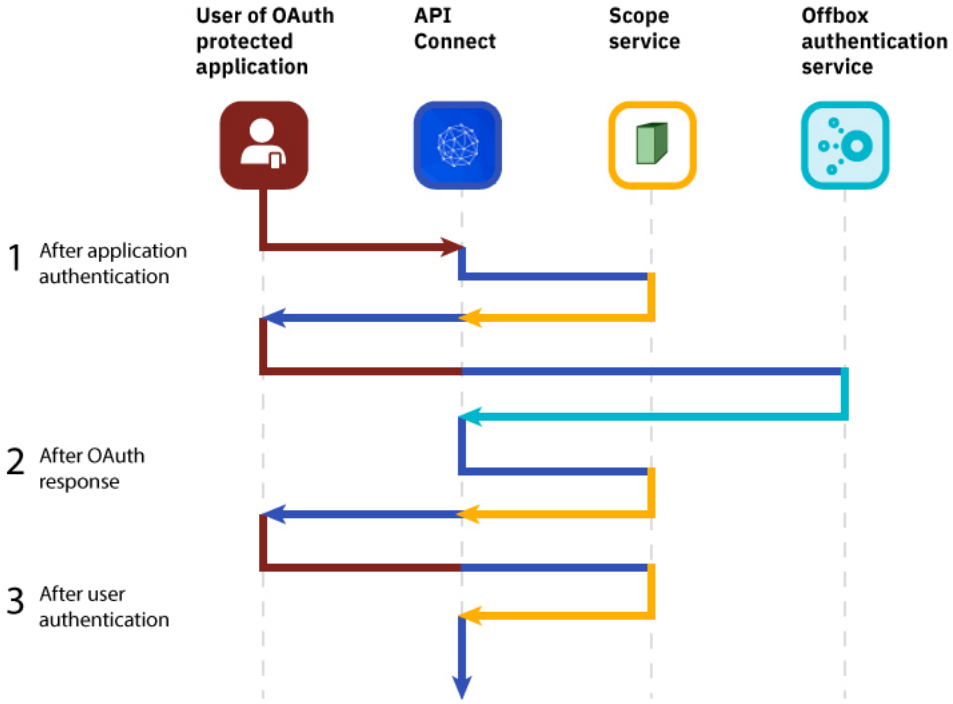
To provide more refined support for the OAuth scope handling, API Connect allows the Authentication URL user registry extension to modify the scope value.

When you define an OAuth provider, the Advanced scope check extensions provide the flexibility to check and override allowed scopes. The optional extensions are Application scope check and Owner scope check.

The scope that is eventually received by the application is determined by the interactions that are described in the three following paragraphs. Scope processing follows the sequence of items 1, 2, and 3 in the following list, offering three opportunities to override the scope value. Figure 1 provides an overview of the process.

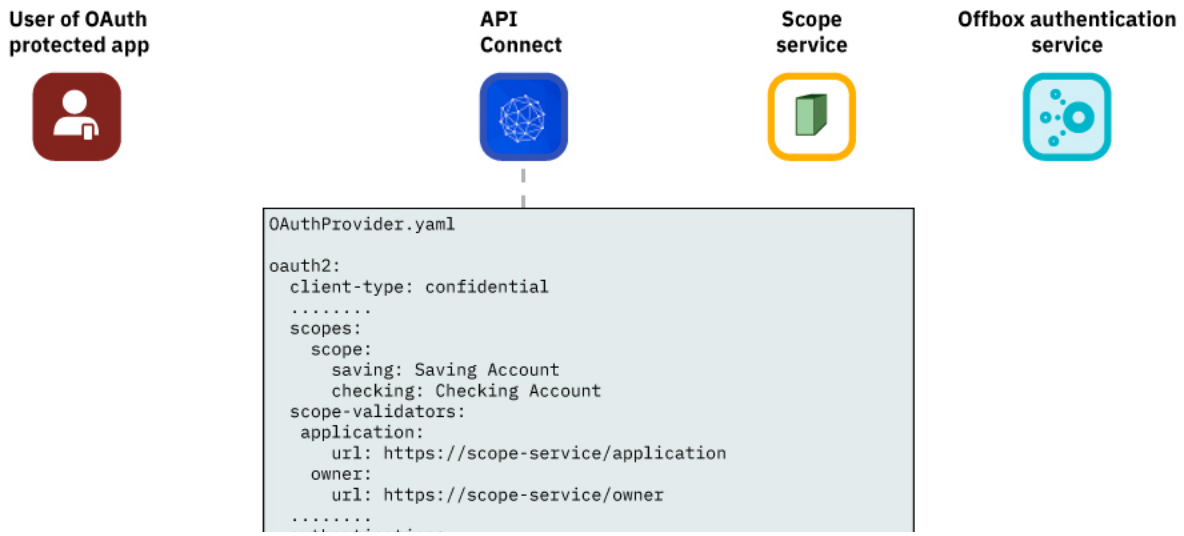
1. After the application successfully authenticates, and if OAuth Native provider > Advanced scope check > Application scope check is configured, API Connect makes a call to allow extra verification and uses the contents of **`x-selected-scope`** to override the scope value that was initially requested by the application. When Application scope check is enabled, the HTTP response header **`x-selected-scope`** must be present, or the call fails.
2. If OAuth Native provider > User Security > Authentication is configured to authenticate application users using an Authentication URL, then API Connect makes a call, as documented in Authentication URL user registry. When the response code is `HTTP 200`, and the response header **`x-selected-scope`** is present, the value that is configured in **`x-selected-scope`** is used as the new scope value, overriding both what the application already requested and what was provided in the Application scope check described in paragraph 1. In the response header, **`x-selected-scope`** is an optional element.
3. After the user successfully authenticates, and if OAuth Native provider > Advanced scope check > Owner scope check is enabled and configured with a valid URL, API Connect makes a call to allow the content of **`x-selected-scope`** to refine the scope value. When Owner scope check is enabled, the HTTP response header **`x-selected-scope`** must be present, or the call fails.

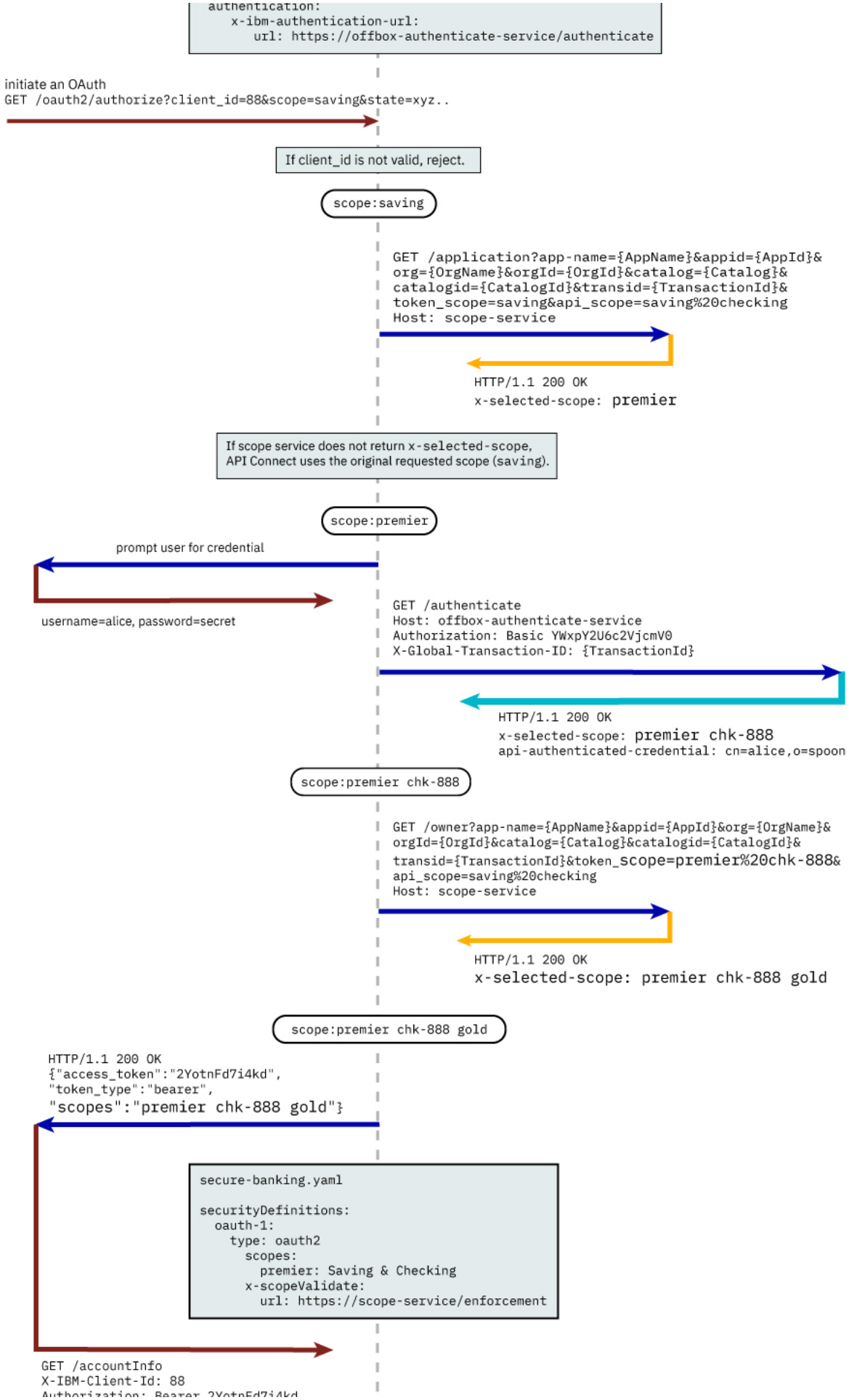
Figure 1. OAuth advanced scope overview

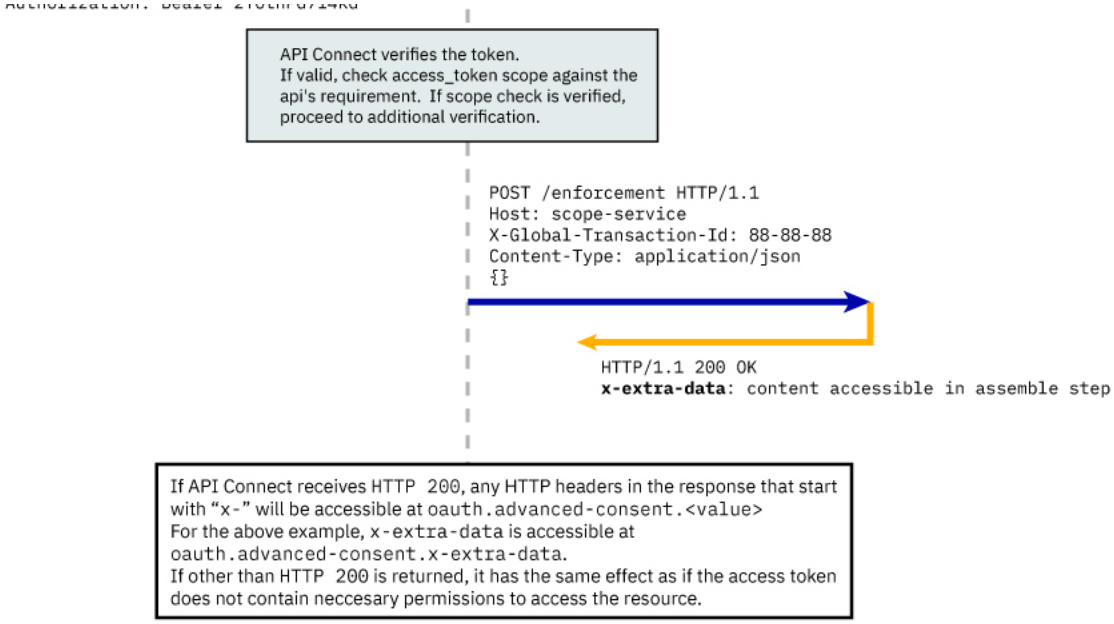


The final scope permission that is granted by the access token is the result of the flow described in items 1, 2, and 3. Figure 2 shows a more detailed view of the transaction flow with examples that show when **`x-selected-scope`** provides a new scope value.

Figure 2. OAuth advanced scope detail




authentication:
x-ibm-authentication-url:
url: https://offbox-authenticate-service/authenticate
initiate an OAuth
GET /oauth2/authorize?client_id=88&scope=saving&state=xyz..
If client_id is not valid, reject.
scope:saving
GET /application?app-name={AppName}&appid={AppId}&
org={OrgName}&orgId={OrgId}&catalog={Catalog}&
catalogid={CatalogId}&transid={TransactionId}&
token_scope=saving&api_scope=saving%20checking
Host: scope-service
HTTP/1.1 200 OK
x-selected-scope: premier
If scope service does not return x-selected-scope,
API Connect uses the original requested scope (saving).
scope:premier
prompt user for credential
username=alice, password=secret
GET /authenticate
Host: offbox-authenticate-service
Authorization: Basic YWxpY2U6c2VjcmV0
X-Global-Transaction-ID: {TransactionId}
HTTP/1.1 200 OK
x-selected-scope: premier chk-888
api-authenticated-credential: cn=alice,o=spoon
scope:premier chk-888
GET /owner?app-name={AppName}&appid={AppId}&org={OrgName}&
orgId={OrgId}&catalog={Catalog}&catalogid={CatalogId}&
transid={TransactionId}&token_scope=premier%20chk-888&
api_scope=saving%20checking
Host: scope-service
HTTP/1.1 200 OK
x-selected-scope: premier chk-888 gold
scope:premier chk-888 gold
HTTP/1.1 200 OK
{"access_token":"2YotnFd7i4kd",
"token_type":"bearer",
"scopes":"premier chk-888 gold"}
secure-banking.yaml
securityDefinitions:
oauth-1:
type: oauth2
scopes:
premier: Saving & Checking
x-scopeValidate:
url: https://scope-service/enforcement
GET /accountInfo
X-IBM-Client-Id: 88



Consumer API enforcement

Standard scope validation
To access the API /getaccount the application must send a **GET** request with an access token that contains the scope, or scopes, defined in the OAuth provider. If the request doesn't contain a scope, you must specify a default scope to use. If no default scope is defined, and the request doesn't contain a scope, an invalid scope error is returned for the request.

```
GET /getaccount
HTTP/1.1
Host: server.example.com
X-IBM-Client-Id: 8888-8888-8888
Authorization: Bearer AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ
```

The following application OpenAPI file secure-banking.yaml defines the scope, or scopes, that must exist in the token to be granted access to the API /getaccount.

secure-banking.yaml

```
securityDefinitions:
   scope-only:
    type: oauth2
    description: ''
    flow: implicit
    authorizationUrl: ''
    scopes:
      checking: 'Checking Account'
      saving: 'Saving Account'
      mutual: 'Mutual Fund Account'
security:
  - scope-only:
      - checking
  - scope-only:
      - saving
      - mutual
```

In the examples, the access token **AAEkNjVkOWIyYjgtOWY5ZS00YWQwLWIyYzktZ** is able to access the API because it contains one, or a combination of **scope-only** that is defined in secure-banking.yaml such as:

- **checking**
- **saving mutual**
- **checking saving mutual**

Advanced Scope Check
Administrators can enable an additional scope check by configuring the consumer API property Advanced Scope Check URL that becomes **x-scopeValidate** as shown in the following OpenAPI file example.

```
securityDefinitions:
  advanced-scope-only:
    type: oauth2
    description: ''
    flow: implicit
    authorizationUrl: ''
    scopes:
      checking: 'Checking Account'
      saving: 'Saving Account'
      mutual: 'Mutual Fund Account'
    x-scopeValidate:
      url: 'https://advanced-scope-check.bk.com/validate-scope'
      tls-profile: 'ssl-client'
```

After API Connect successfully verifies the access token against any scope requirement, API Connect will make an **`HTTP POST`** request to the **`x-scopeValidate`** endpoint similar to the following example. Response code `HTTP 200` from the endpoint indicates a success. Any other response code, or a connection error, is treated as a permission error for the token.

```
     POST /validate-scope?app-name=..&appid=..&org=..&orgid=..&catalog=..&catalogid=..&transid=..
HTTP/1.1
     Host: advanced-scope-check.bk.com
     Content-Type: application/json

   {"context-root" : checking,
    "resource" : accountinfo,
    "method" : GET,
    "api-scope-required" : [jointaccount],
    "access_token" : {"client_id" : "2cd71759-1003-4a1e-becb-0474d73455f3",
                      "not_after" : 174364070,
                      "not_after_text" : "2017-07-11T02:27:50Z",
                      "not_before" : 174360470,
                      "not_before_text" : "2017-07-11T01:27:50Z",
                      "grant_type" : "code",
                      "consented_on" : 1499736470,
                      "consented_on_text" : "2059-07-11T01:27:50Z",
                      "resource_owner" : "cn=spoon,email=spoon@poon.com",
                      "scope" : "jointaccount mutual",
                      "miscinfo" : "[r:gateway]"
                     }
}
```

An example of successful response follows.

```
     HTTP/1.1 200 OK
     Cache-Control: no-store
     Pragma: no-cache
     X-Custom-For-Assemble-Process: audit
```

Any HTTP response header that begins with "**`x-`**" is kept as the context variable **`oauth.advanced-consent`**. Based on the example successful response, **`X-Custom-For-Assemble-Process: audit`** becomes `oauth.advanced-consent.x-custom-for-assemble-process`, and can be accessed in the assemble step.

Additional validation options
You can optionally use additional fields for validation:

Request Header
: Defines the regular expression to match against **request** headers. Matching headers are included in the request to the advanced scope validation endpoint.

Response Context Variable
: Defines the regular expression to match against **response** headers. Matching headers are saved as context variables in the format **`oauth.advanced-consent.*`**.

Related information

- IETF RFC 6749

Tokens

Tokens can be managed using refresh tokens and revocation URLs. You can extend the life of tokens using refresh tokens. You can end the life of a token by specifying a revocation URL.

This section contains information regarding token management, including refreshing and revoking tokens, redirecting, and managing tokens with the DataPower Gateway.

- **Refresh tokens**
  If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.
- **Token revocation**
  In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.
- **Authenticating and authorizing through a redirect URL**
  You can use a service that is hosted externally from IBM API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.
- **Token management with the DataPower Gateway (v5 compatible)**
  API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.
- **Token management with the DataPower API Gateway**
  API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Refresh tokens

If you are using OAuth authentication, you can enable refresh tokens. Refresh tokens are issued to the client to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope.

When you are using OAuth authentication, API requests must include a valid access token, by using the Authorization HTTP header. Access tokens that are issued by the IBM® API Connect Token Endpoint are valid for 3600 seconds (1 hour) by default, as indicated by the expires_in property that is returned on the token request. The following code block shows an example API request with an Authorization header:

```
GET /bankingApi/accountSummary?client_id=32427ce5-bb7c-48a7-9de3-4bb629091103
```

```
HTTP/1.1
Accept: application/json
Host: api.ibm.com
Authorization: Bearer AAEFYy1hbGxlhdS5nVX4x6iTL2sb3ymBivQb...
```

After an access token expires, if the option is enabled in the OAuth provider configuration Tokens > Refresh tokens screen, the application uses refresh tokens. Each refresh token is valid for approximately 31 days after it is issued (or for the Time to Live time period specified) and can be used only once to request a new access token. Along with the new access token, a new refresh token is also returned. For details on how to enable refresh tokens, see Configuring a native OAuth provider.

If the access token is expired and the application does not have a refresh token, it must restart the OAuth exchange by using the choice of Grant Type(s) allowed by the OAuth provider.

Token revocation

In IBM® API Connect, you can revoke or refresh tokens. If necessary, you can also revoke all tokens that are issued to a specific client ID or a resource owner.

Token revocation using an external third party service

This topic describes token revocation using a third party, external service. This option is configured in the Native OAuth provider, using the Token Management > Type = Third party screen. The revocation URL is an endpoint that links to an external service which contains information about access or refresh tokens. API Connect is involved in the initial creation and validation of tokens. When an OAuth revocation URL is present, API Connect calls the URL to determine if the associated token can be trusted. The token server then checks a token *blocklist* (a data store of inactive tokens) to ensure that the token is still valid. If the token is still valid, API Connect continues the processing.

Examples

A number of revocation examples follow. The first shows a sample fetch request and the response from a remote revocation URL.

GET request and response

In this example, an API Gateway server issues a GET request to the Revocation URL and receives a result. It shows that different resource owners (Laura and Emily) can revoke all tokens when using the same application (client ID).

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
GET <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Accept: application/xml; text/xml
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>

<!--
Access Tokens and/or Refresh Tokens that are revoked can be individually listed.
To keep this list small, please only include access tokens and refresh tokens that are valid.
For access tokens, any token older than 20 minutes is no longer valid.
For refresh tokens, any token older than 44700 minutes is no longer valid.
-->

<token type="access">AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1...</token>
<token type="refresh">fZaRlVbnPSc1UGTjCRdq4mPbOosD2+aZIKbJ6bTeW...</token>

<!--
If a resource owner has revoked all tokens issued to a given application, please
list them as shown here.
-->

<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">laura</resource-owner>
<resource-owner client-id="760d75a2-44b1-4485-8c6f-0d264fcf7398">emily</resource-owner>

</oauth-revocation>
```

POST request and response

The next example shows a post request and response.

Request:

```
POST <revocationURL> HTTP/1.1
User-Agent: IBM-APIManagement/4.0
Content-Type: application/xml

<?xmlversion="1.0" encoding="UTF-8"?>
<token>
   <token_type>bearer</token_type>
   <access_token>AAENYy1hbGwtcmVmcmVzaOfNeQKX8ZeojsBY9v0FI7/OerQvzKHq...</access_token>
   <expires_in>3600</expires_in>
   <scope>3600</scope>
   <resource-owner>alice</resource-owner>
   <client_id>83d9cdcd-ba72-4d00-abae-005da8da5fb1</client_id>
</token>
```

Response:

```
HTTP/1.1 200 OK
```

Provide token information on revocation request

In this example, the application calls an API and passes a bearer token. In response, the Gateway fetches the revocation URL and provides information on the token being verified.

Gateway:

```
GET  <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Alternatively, the same process occurs when using a refresh token to issue a new access token. The application sends a refresh request to the token service. The Gateway then fetches the revocation URL, providing information on the refresh token being verified.

Gateway:

```
GET  <revocationURL>GET  HTTP/1.1
refresh-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Note: If you are using a third party OAuth provider then, for API Calls with bearer tokens, when Introspect URL is enabled on the API, the revocation URL is not applicable. Instead, the third party endpoint must validate the token and also check for revocation before returning a `200 OK` response to the Gateway.

Revoking tokens issued to Alice up to and including a specific date

On May 1st, Alice loses her phone and needs to reset her password. As a result, the token provider wants to revoke every token issued before Alice lost her phone. In this example, the Gateway sends a `GET` request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
  <resource-owner before="2015-05-01T09:30:10Z">alice</resource-owner>
<oauth-revocation>
```

Revoking all tokens issued up to and including a specific date

Under certain catastrophic conditions, you may need to revoke all tokens issued up to and including a specific date, for example, May 1st. In this example, the Gateway sends a `GET` request to the revocation service. The revocation service replies confirming revocation of the specified tokens.

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-revocation>
  <everytoken before="2015-05-01T09:30:10Z" />
<oauth-revocation>
```

Putting it all together

The following shows the examples contained in this topic executed in a single action:

Gateway:

```
GET <revocationURL>HTTP/1.1
access-token: AAETb2F1dGgtcmV2b2tlLWN1c3RvbfZaRlVbnPSc1
client-id: 760d75a2-44b1-4485-8c6f-0d264fcf7398
resource-owner: alice
```

Revocation Service:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Cache-Control: public, max-age=120
Date: Fri, 08 May 2015 21:49:03 GMT


<?xml version-"1.0" encoding="UTF-8"?>
```

```
<oauth-revocation>
  <resource-owner before="2015-04-08T09:30:10Z">mary</resource-owner>
  <resource-owner before="2015-04-12T09:30:10Z">john</resource-owner>
  <resource-owner before="2015-04-13T09:30:10Z">kevin</resource-owner>
  <resource-owner before="2015-04-01T09:30:10Z">alice</resource-owner>
<oauth-revocation>
```

Notes:

- In the previous example, there are no entries older than one month in the response (the maximum life of a refresh token).
- Each response is cached for up to two minutes according to response's directive.
- The **`before`** attribute uses the **`xs:dateTime`** syntax.

Authenticating and authorizing through a redirect URL

You can use a service that is hosted externally from IBM® API Connect to collect authentication and authorization details from your user when an application requests access on that user's behalf.

Before you begin

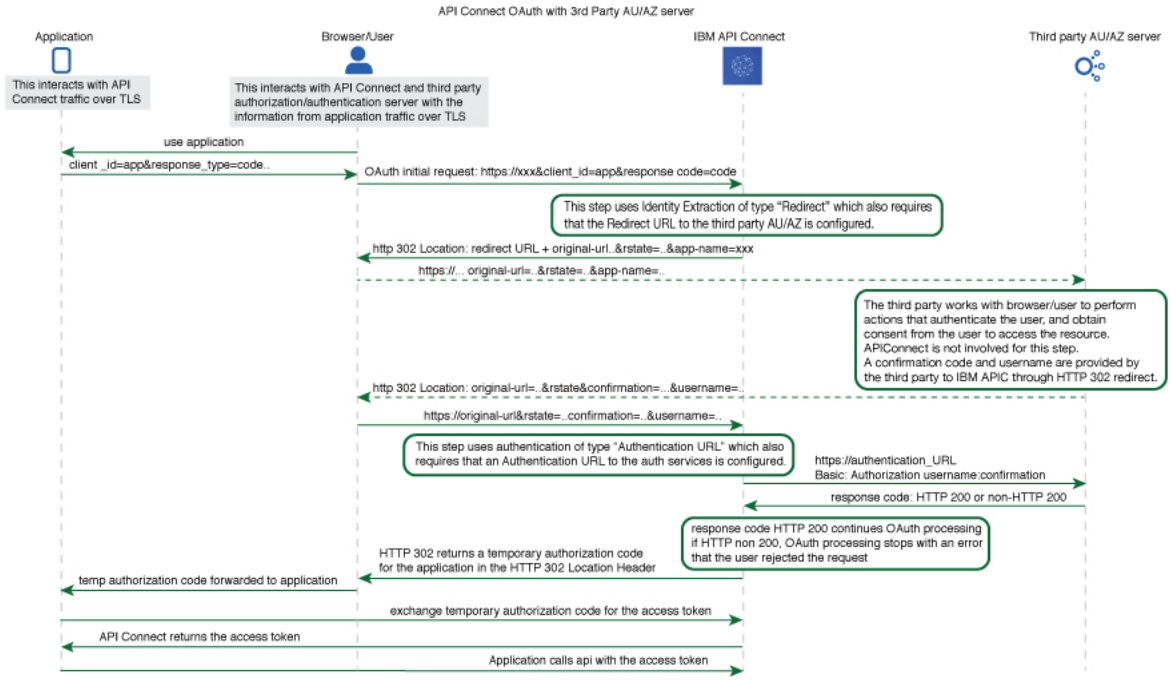
To complete this task, you will need to either create or have created an OAuth security definition that uses Implicit grant type or Access Code (Authorization Code) grant type. For more information, see Creating an OAuth security definition.

About this task

If you use methods for authentication that are not supported by API Connect, you can redirect users to a suitable URL at which they can authenticate. The user is then returned to the OAuth process after authentication and authorization have been confirmed.

The following illustration indicates the transaction flow for third party authentication.

Figure 1. Third party authentication (AU) and authorization (AZ) transaction flow



1. The application initiates a request to access an API protected with a third-party entity. API Connect redirects the application with an **`HTTP 302`** redirect based on **`identity extraction -> redirect -> redirect-url`**, for user authentication (and optional authorization).
2. The application communicates directly with the third-party entity to gather user identity. API Connect is not involved in this communication. After the third-party entity finishes processing authentication (and optional authorization), it returns an **`HTTP 302`** redirect that uses the **`original-url`** from the request, with the username and confirmation code appended.
3. API Connect receives the request that includes the username and confirmation code, and communicates with the authentication URL, based on **`authentication -> x-ibm-authentication-url`**, to confirm user identity before the request is completed.
4. An **`HTTP 200`** response from the third-party entity allows API Connect to continue the OAuth 2.0 request process as if the owner is authenticated. The request is then processed according to the **`grants`** type.
   - **`- accessCode`** returns a temporary code to the application.
   - **`- implicit`** returns the access token to the application.

   For any response other than HTTP 200, the request fails with a statement added to the error log.

Procedure

To create an external form, and to indicate the URL to which API Connect will redirect users, complete the following instructions:

1. Create your service for authentication and authorization. You will use the URL of the landing page of this service as your redirect URL.
   a. To include elements in your form that are provided by API Connect, use the following query parameters from the URL that your user is redirected to.
   Note: With the exception of **`original-url`**, none of these parameters are included in the URL automatically; you must add them as required.

      app-name
      : The name of the application requesting access, as provided through the Developer Portal.

      appid
      : The id of the application requesting access.

      catalog
      : The name of the catalog where the product is being used by the application.

      catalogid
      : The id of the catalog where the product is being used by the application.

      catalogtitle
      : User-friendly display name for the catalog.

      org
      : The name of the consumer organization that hosts the application.

      orgid
      : The id of the consumer organization that hosts the application.

      orgtitle
      : User-friendly display name for the organization.

      original-url
      : The original URL that the user was directed to by the application, including query parameters from the original URL that are necessary for standard OAuth 2.0 requests. You can include these parameters in your service to provide information to the user. Additionally the state_nonce is appended. The state_nonce is a hash code generated by API Connect for verification purposes. The URL is URL-encoded and should be decoded before further use, the **`state_nonce`** should remain unchanged.

      provider
      : The name of the API provider organization.

      providerid
      : The id of the API provider organization.

      providertitle
      : User-friendly display name for the provider organization.

      requested-scope
      : [optional] If Application Scope check is enabled and replaces the **`scope`** from the initial application request, this field holds the **`scope`** value from the initial application request, and the new replacement scope value is put into **`original-url`**.

      transid
      : transaction id used in the GW for the transaction which trigger this call

      The URL to which the user is sent to when they are redirected to your page has the following form:

      ***`Redirect_URL`*`?original-url=`*`Original_URL`*`&state_nonce=`*`R_State`*`&app-name=`*`Application_Name`**

      where all variables are as described previously. The Redirect URL does not have a size limit enforced by API Connect.

   b. Create the stages of authentication, authorization, and any intermediate stages that you require to take place before you allow access to the application. Upon completion of these stages, redirect the user to the *Original_URL* and append a user name, their confirmation code, and the application name to be evaluated for access grant or denial by API Connect. The confirmation code does not have a size limit enforced by API Connect.
   Original URL requires the following form:

      ***`Original_URL`*`&username=`*`User_Name`*`&confirmation=`*`Confirmation_Code`**

      where all variables are as described previously.

      For example:

      ```
      https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
      response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
      5709432af66e&state_nonce=HoIbRG+6bZtqlB7LDkq4gjlD3SHKglCbnYdHs/bMz2Y=&username=spoon&confirmation=12345678
      ```

   c. To send your own error responses after the authentication and authorization service, redirect the user to the *Original_URL* and append an error code. You can also append a user name. Use the following form:

      ***`Original_URL`*`&username=`*`User_Name`*`&error=`*`Error_Response`**

      where *Error_Response* is the message you wish to send and all other variables are described as previously.

      For example:

      ```
      https://your_gateway.com/your_org/your_catalog/your_api/oauth/authorize?
      response_type=code&redirect_uri=https://example.com/redirect&scope=/your_api&client_id=5af57a4a-6db9-4141-ad08-
      5709432af66e&state_nonce=HoIbRG+6bZtqlB7LDkq4gjlD3SHKglCbnYdHs/bMz2Y=&username=User&error=access_denied
      ```

2. Create a service to validate the confirmation code and user name. API Connect makes a GET call to your authentication URL after the user is redirected back to the authorization URL. When the call is made, it includes in its authorization header the user name and confirmation code you supplied previously. Confirm that these are correct and respond with an HTTP success code such as `200 OK` if you want to allow access, or non-`200 HTTP` response code, such as `401 Unauthorized` to deny access.
3. In your OAuth provider configuration, supply the redirect URL that is used in Step 1 and the authentication URL that is used in Step 2.
   For more information on configuring an OAuth Provider, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Token management with the DataPower Gateway (v5 compatible)

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

In order to manage tokens with the DataPower® Gateway (v5 compatible), you must set the Token Management Type to Native in your Native OAuth provider configuration. See Configuring token management and revocation for a native OAuth provider when using Cloud Manager or Configuring token management and revocation for a native OAuth provider when using API Manager.

Also, the DataPower quota enforcement server must be enabled on the DataPower Gateway to use the distributed cache to manage tokens. See Quota enforcement.

Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

When distributed cache support is enabled, replay protection is provided across the gateway cluster through the quota enforcement server. This support ensures that the same token cannot be reused across the members of the quota enforcement peer group.

Important:
If you have a cluster of DataPower Gateway servers, the OAuth data synchronization behavior across the servers depends on whether or not you enable revocation:

- If you enable revocation, API Connect uses the DataPower quota enforcement server, and OAuth data is synchronized across the servers. If an access token is obtained from one server, the OAuth data synchronization ensures that the same authorization code cannot be used to obtain an access code from another server. You must ensure that the DataPower quota enforcement server is configured.
- If you disable revocation, API Connect does **not** use the DataPower quota enforcement server and OAuth data does not synchronize across the cluster of DataPower Gateway servers. Therefore, to prevent the same authorization code being used to obtain an access code from more than one server you must configure DataPower to synchronize OAuth data across the servers, by using the DataPower Gateway console.

To configure DataPower to synchronize OAuth data, ensure that the DataPower quota enforcement server is configured. For more information, see Configuring the quota enforcement server.

Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user `cn=spoon,o=ibm` with username `spoon` and password `spoon` using a registered administration application of `5287fe53-8747-438a-8262-681ec75b79c5`.

- Request:

```
GET /oauth2/issued
HTTP/1.1
       Host: apic.ibm.com
       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
       Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

[

{

"clientId": "5287fe53-8747-438a-8262-681ec75b79c5", => same client id as request

"owner": "cn=spoon,o=ibm",

"clientName": "PetStore Application",

"scope": "listpet",

"issuedAt": 1503327054,

"consentedOn": 1503327054,

"expiredAt": 1503330654,

"refreshTokenIssued": false,

"appId": "d2031f0f27339315333734ab9",

"org": "PetStoreOrg",

"orgTitle": "Katie Pet Grooming Inc",

"orgId": "5887803de4b06e6998c4b2c7",

"provider": "SuperStore",

"providerTitle": "Simon SuperStore",
```

```
"providerId": "5887803de4b06e6998c4b2c7",

"catalog": "publicapi",

"catalogTitle": "For public",

"catalogId": "5887803de4b06e6998c4b2d3"

} => delete second item

]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8

HTTP/1.1

Host: apic.ibm.com

x-ibm-client-id: a8746323-9825-a842-8736-abd8202356ac8 => same client id as delete request

x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7

Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

    { "status": "success" }
```

Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 IETF RFC 7009. An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded

     token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn94lSWX14xuR....
```

- Response:

```
HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: private, no-store, no-cache, must-revalidate
     Pragma: no-cache

    { "status": "success" }
```

Revoke `refresh_token`

- Request:

```
POST /oauth2/revoke
HTTP/1.1

   Host: apic.ibm.com

      x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
      x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
     Content-Type: application/x-www-form-urlencoded

     token_type_hint=refresh_token&token=........
```

- Response:

```
HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: private, no-store, no-cache, must-revalidate
     Pragma: no-cache

    { "status": "success" }
```

Token management with the DataPower API Gateway

API Connect can use the DataPower distributed cache to manage the token lifecycle that includes when to revoke access rights.

Note: The steps described in this topic are required if you are using the DataPower® OVA or appliance version of the gateway. For the API Connect supplied Kubernetes deployment, the API Security Token Manager is preconfigured provided that, in the gateway service custom resource file, v5 compatibility mode is disabled and the token management service is enabled:

```
      .
      .
      .
spec:
      .
      .
      .
  apicGatewayServiceV5CompatibilityMode: false
  tokenManagementService:
    enabled: true
      .
      .
      .
```

For more information, see Installing the DataPower Gateway subsystem.
Token management relies on gateway-peering to distribute the cache for revocation details within a gateway cluster node, and does not propagate across different gateway clusters. In order to enforce token management across different gateway clusters, you must use the external token store and set the Token Management Type to **External** in your native OAuth provider configuration.

In order to manage tokens with the DataPower API Gateway, you must set the Token Management Type to Native in your Native OAuth provider configuration. See Configuring token management and revocation for a native OAuth provider when using Cloud Manager or Configuring token management and revocation for a native OAuth provider when using API Manager.

Note: This setting is also required when you use a third-party OAuth provider and **`Introspect cache type`** is set to **`protocol`** or **`Time to live`**.
For token management with the DataPower API Gateway, you must configure the API Security Token Manager on the gateway. Complete the following steps:

1. Log in to the DataPower administration console, selecting apiconnect for the domain, and WebGUI for the Graphical Interface.
2. In the search box, enter API Security Token Manager, then click on the API Security Token Manager link that displays in the search results.
3. For the Administrative state, select enabled.
4. Click the + icon alongside the Gateway Peering field to create a new gateway peering object.
5. Provide a Name and a Local address.
6. In the Local port and Monitor port fields, provide values that aren't already in use by services for the Local address.
7. Ensure that Peer group mode is selected.
8. Alongside the Peers field, use the add button and accompanying field to add at least two peers, to ensure that quorum is achieved.
9. If the Enable SSL option is selected, select a value for the Identification credentials.
10. For Persistence location, select a setting other than memory.
11. When done, click Apply.
12. Repeat steps 1 to 11 for each DataPower API Gateway in the peer group.
13. When done, click Apply, then click Save Configuration to save your changes.

For more information, see Defining the API security token manager and Creating a gateway peering instance.

Resource owner revocation

When Resource owner revocation path is selected in the Token Management screen, the configuration inserts two REST API calls to /oauth2/issued.

- An HTTP GET operation that retrieves a list of all granted permissions for a specific user.
- An HTTP DELETE operation that revokes an application for a specific user.

The setting inserts header-based security definitions of client ID and client secret as shown in the **View permissions example**. The API call to revoke a given application for a given user is shown in the **Revoke permissions example**.

View permissions example

To list all the applications granted by user **`cn=spoon,o=ibm`** with username **`spoon`** and password **`spoon`** using a registered administration application of **`5287fe53-8747-438a-8262-681ec75b79c5`**.

- Request:

```
GET /oauth2/issued
HTTP/1.1
       Host: apic.ibm.com
       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
       Authorization: Basic c3Bvb246c3Bvb24=
```

- Response:

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

[
```

```
{
"clientId": "5287fe53-8747-438a-8262-681ec75b79c5", => same client id as request
"owner": "cn=spoon,o=ibm",
"clientName": "PetStore Application",
"scope": "listpet",
"issuedAt": 1503327054,
"consentedOn": 1503327054,
"expiredAt": 1503330654,
"refreshTokenIssued": false,
"appId": "d2031f0f27339315333734ab9",
"org": "PetStoreOrg",
"orgTitle": "Katie Pet Grooming Inc",
"orgId": "5887803de4b06e6998c4b2c7",
"provider": "SuperStore",
"providerTitle": "Simon SuperStore",
"providerId": "5887803de4b06e6998c4b2c7",
"catalog": "publicapi",
"catalogTitle": "For public",
"catalogId": "5887803de4b06e6998c4b2d3"
} => delete second item
]
```

Revoke permissions example

To revoke application `a8746323-9825-a842-8736-abd8202356ac8` by owner `cn=spoon,o=ibm`.

- Request

```
DELETE /oauth2/issued?client-id=a8746323-9825-a842-8736-abd8202356ac8
HTTP/1.1
Host: apic.ibm.com
x-ibm-client-id: a8746323-9825-a842-8736-abd8202356ac8 => same client id as delete request
x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
Authorization: Basic c3Bvb246c3Bvb24=
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: private, no-store, no-cache, must-revalidate
Pragma: no-cache

     { "status": "success" }
```

Client revocation

When Client revocation path is selected in the Token Management screen, the result is the following:

This option inserts one REST API call to /oauth2/revoke, which supports OAuth 2.0 IETF RFC 7009. An HTTP POST operation that an application can send to this API to revoke either an `access_token`, or `refresh_token` with `token_type_hint` as shown in the following examples:

Revoke `access_token`

- Request:

```
POST /oauth2/revoke
HTTP/1.1

    Host: apic.ibm.com

       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
      Content-Type: application/x-www-form-urlencoded

      token_type_hint=access_token&token=AAIHZGVmYXVsdD1-KqwD0Yc3EDn94lSWX14xuR....
```

- Response:

```
HTTP/1.1 200 OK
      Content-Type: application/json;charset=UTF-8
      Cache-Control: private, no-store, no-cache, must-revalidate
      Pragma: no-cache

     { "status": "success" }
```

Revoke `refresh_token`

- Request:

```
POST /oauth2/revoke
HTTP/1.1

    Host: apic.ibm.com

       x-ibm-client-id: 5287fe53-8747-438a-8262-681ec75b79c5
       x-ibm-client-secret: E2qM6mG2bX2uC1xT2iN1uU6bT5cV4dN7nW5kM5uP8vL3uF3cT7
      Content-Type: application/x-www-form-urlencoded

      token_type_hint=refresh_token&token=........
```

- Response:

```
HTTP/1.1 200 OK
      Content-Type: application/json;charset=UTF-8
      Cache-Control: private, no-store, no-cache, must-revalidate
      Pragma: no-cache

     { "status": "success" }
```

Authentication URL user registry

You can use an Authentication URL user registry to specify a REST authentication service that manages user authentication, and optionally provides additional metadata to be embedded in the token.

This support can optionally enable any of the following:

- Providing the authenticated credential to IBM® API Connect. For example, the user logs-in with user name: `spoon`, and password: `fork`. When the user is authenticated, the credential becomes **`cn=spoon,o=eatery`**. The credential is kept in the OAuth **access_token** to represent the user.
- Providing metadata support. Allow extra metadata to be stored in the **`access_token`**.
- Overriding the **`scope`** that the application receives after a successful OAuth protocol processing. By responding with a specific header, the Authentication URL endpoint can replace the **`scope`** value that the application receives. For example, you can provide a specific resource owner an account number within the **`scope`** header response for use in future processing steps.

When you call the Authentication URL user registry, the API Connect gateway sends a GET request with HTTP headers and then processes any HTTP response from the URL. For authentication, a REST authentication service is expected at the Authentication URL.

The following response from the REST authentication service indicates that user authentication is successful and that API Connect will use **`cn=spoon,o=eatery`** as the user identity.

```
HTTP/1.1 200 OK
Server: example.org
X-API-Authenticated-Credential: cn=spoon,o=eatery
```

For information on how to configure a User Security policy in an API assembly for use with an Authentication URL user registry, see User Security policy.

For an example of an OAuth provider configuration that uses an Authentication URL user registry, see Example - using multiple OAuth policies in an OAuth provider assembly.

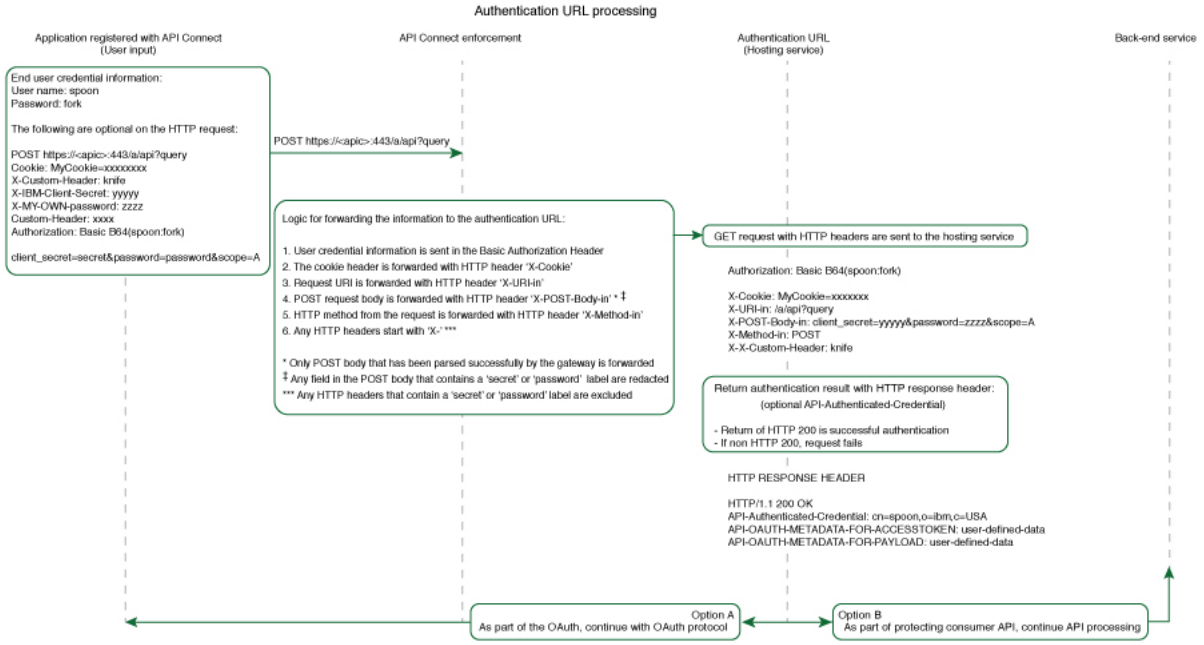
API Connect considers any non-200 HTTP response code a failed user authentication attempt.

When an Authentication URL user registry is invoked, two HTTP response headers are available that include metadata in the access token or the response payload that contains the access token. For more information, see OAuth external URL and authentication URL. The two metadata response headers are:

```
API-OAUTH-METADATA-FOR-ACCESSTOKEN
API-OAUTH-METADATA-FOR-PAYLOAD
```

When an Authentication URL is invoked, an HTTP response header is available to override the requested **`scope`** from the application. For more information, see Scope. The response header is:

```
x-selected-scope
```



If you are using the DataPower® API Gateway rather than the DataPower Gateway (v5 compatible), this diagram is provided for guidance only and is not fully accurate for this release.

Custom forms for user security

You can create custom forms for the authorization and identity extraction phase of OAuth.

About this task

The Native OAuth provider configuration provides the capability for requiring additional authentication and authorization steps for user security. Custom HTML forms may be created to extract the identity of the user and to authorize the user. This capability applies to Implicit, Resource owner password, and Access code grant types.

- **Creating a custom HTML login form for user security**
  Custom HTML forms can be created for user security during the identity extraction stage in OAuth.
- **Creating a custom HTML authorization form for user security**
  Custom HTML forms can be created for user security during the authorization stage in OAuth.

Creating a custom HTML login form for user security

Custom HTML forms can be created for user security during the identity extraction stage in OAuth.

Before you begin

The Native OAuth provider configuration includes Identity Extraction when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to extract the user credential and one of the choices is Custom HTML Form. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager. For more information, see Configuring a native OAuth provider. This topic describes how to create the Custom HTML form for identity extraction.

About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form for signing in to the service provided by the API. You can present a custom form or a default form. Your custom form must fulfill certain requirements.

Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

Procedure

To create a custom sign-in form for your Native OAuth provider, complete the following steps:

1. Create a well formed XHTML document that will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as `POST`, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.
   For example:

```
<form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
```

3. Create a text input field that is named `username` and create a password input field named `password`.
4. Add the line `<EI-INJECT-HIDDEN-INPUT-FIELDS>`. This third element is a placeholder that API Connect replaces with input fields to complement the user-submitted data.
5. Create a button to initiate the sign-in process.
   For example:

```
<button
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.apionprem.doc_oauth_custom_login_form_2_login_
button" type="submit" name="login" value="true">Log in</button>
```

6. Optional: Add text that is displayed the first time that the user visits the sign-in page. Use the tag `<EI-LOGINFIRSTTIME>` for the text that you want to display.
7. Optional: Add text that appears when the user is returned to the sign-in page if they fail to authenticate. Use the tag `<EI-LOGINFAILED>` for the text that you want to display.
8. Optional: Have an error message displayed when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<EI-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
9. Optional: You can add elements that are loaded from external sources, such as images or JavaScript.
10. Insert spacing and other features as you require. Completing Steps 1 through to 8 results in a form similar to the following example:

```
<html lang="en" xml:lang="en">
  <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/></head>
  <body>
    <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
      <h1>Please sign in</h1>
      <p>Username </p>
      <p ><input type="text" name="username" required="required" /> </p>
      <p>Password </p>
      <p ><input type="password" name="password" required="required" /> </p>
      <EI-INJECT-HIDDEN-INPUT-FIELDS/>
      <p > <button
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.apionprem.doc_oauth_custom_login_form_2_login_
button" type="submit" name="login" value="true">Log in</button> </p>

      <EI-LOGINFIRSTTIME>
        <p>If you have forgotten your user name or password, contact your system administrator.</p>
      </EI-LOGINFIRSTTIME>

      <EI-LOGINFAILED>
        <p >At least one of your entries does not match our records.
          If you have forgotten your user name or password, contact your system administrator.</p>
      </EI-LOGINFAILED>

      <EI-INTERNAL-CUSTOM-FORM-ERROR/>

    </form>
  </body>
</html>
```

11. Make your form available at a URL of your choice.
12. If you have not already done so, configure your Native OAuth provider to use a Custom HTML form for identity extraction and provide the URL at which your form is available. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Related tasks

- Creating a custom HTML authorization form for user security

Related information

- Configuring API security

Creating a custom HTML authorization form for user security

Custom HTML forms can be created for user security during the authorization stage in OAuth.

Before you begin

The Native OAuth provider configuration includes user authorization when using the Implicit, Access code, or Resource owner password grant types. You have the option to select how to authorize application users and one of the choices is Custom HTML Form. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager. This topic describes how to create the Custom HTML form for authorization.

About this task

During three-legged OAuth definitions (Implicit flow, Resource owner password flow, and Access (Authorization) code flow, the user is presented with a form through which they grant permission to an application to access their data through the API on their behalf. You can present a custom form or a default form. Your custom form must fulfill certain requirements.
Important: The fields used by IBM® API Connect to inject information into your form have case-sensitive field names.

Procedure

To create a custom authorization form for your Native OAuth provider, complete the following steps:

1. Create a well-formed XHTML document. This will be parsed and transformed by API Connect to inject hidden fields.
2. For your XHTML form, set the method as `POST`, the encoding type as `application/x-www-form-urlencoded`, and the action as `authorize`. Add any other parameters that you require.
   For example:

   ```
   <form method="POST" enctype="application/x-www-form-urlencoded" action="authorize">
   ```

3. Add the line `<AZ-INJECT-HIDDEN-INPUT-FIELDS/>`. This line is a placeholder that API Connect will replace with input fields necessary for the completion of the OAuth process.
4. Create two buttons with the following code so that the user can grant or deny permission. Edit the text to suit your preferences.

   ```
   <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
   <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
   ```

5. Optional: Display an error message when an error in the custom form prevents it from being displayed to the user correctly. Use the tag `<AZ-INTERNAL-CUSTOM-FORM-ERROR/>`; the message text is generated automatically. You should detect such errors during testing to prevent this error message being displayed to the end user.
6. Optional: You can add to the form HTML elements that will load features from external sources, such as images or JavaScript.
   For example, `<script src="http://www.example.com/example.js" />`
7. Insert spacing and additional elements as you require. Completing Steps 1 through to 6 results in a form similar to the following example:

   ```
   <html lang="en" xml:lang="en">
     <head><title>Request for permission</title></head>
     <body class="customconsent">
       <div>
         <div>
           <form method="post" enctype="application/x-www-form-urlencoded" action="authorize">
             <AZ-INJECT-HIDDEN-INPUT-FIELDS/>
             <p>Greeting..</p><DISPLAY-RESOURCE-OWNER/>
             <p>This app </p><OAUTH-APPLICATION-NAME/><p> would like to access your data.</p>
             <div>
               <button class="cancel" type="submit" name="approve" value="false">No Thanks</button>
               <button class="submit" type="submit" name="approve" value="true">Allow Access</button>
             </div>
           </form>
         </div>
         <AZ-INTERNAL-CUSTOM-FORM-ERROR/>
       </div>
     </body>
   </html>
   ```

8. Make your form available at a URL of your choice.
9. If you have not already done so, configure your Native OAuth provider to use a Custom HTML Form for authorization for User Security. Provide the URL as the endpoint at which your form is available. For more information, see Configuring a native OAuth provider when using API Manager or Configuring a native OAuth provider when using Cloud Manager.

Related tasks

- Creating a custom HTML login form for user security

Related information

- Configuring API security

Securing an API with a JSON Web Token

There are two methods to secure your API with a JSON Web Token. You can use the **jwt-generate** command, or you can use a token that has been generated external to IBM® API Connect.

About this task

JSON Web Token (JWT) is an OAuth 2.0 compliant method of authentication that can be useful to secure your API in API Connect.

You can secure your API with a JSON Web Token by using either of the following methods:

- Generate a token through the **jwt-generate** command, and then augment the response payload with your generated token replacing the `id` token.
- Use a token that was generated outside of API Connect and include it into the response payload, by using the metadata URL.

Procedure

Create a **jwt-generate** policy in the assembly.

a. In API Manager, open the Assembly tab.
b. Add a jwt-generate policy to the assembly for the API.

Troubleshooting OAuth

You can find the answers to some common questions in the Developer Portal, or in support communities and forums in DeveloperWorks, on GitHub, or on YouTube.

You can use the following links to go to the topics:

- OAuth 2.0 support in the developer portal test tool
- OAuth requires quorum in a multi-node cluster

OAuth 2.0 support in the developer portal test tool

OAuth 2.0 support is exposed as an API through the provider OpenAPI definition. You can use the test tool that is included with API Connect to test the OAuth 2.0 configuration. For more information, see Testing an API using the Developer Portal test tool.

OAuth requires quorum in a multi-node cluster

In a multi-node cluster, OAuth operations will fail if quorum is lost. Quorum requires that the number of active nodes is greater than 50% of the total number of nodes in the cluster.

API Analytics

You can use API Connect to filter, sort, and aggregate your API event data. You can present the results within correlated charts, tables, and maps to help you manage service levels, set quotas, establish controls, set up security policies, manage communities, and analyze trends.

API analytics is built on the OpenSearch open source real-time distributed search and analytics engine.

The content of the API event data depends on the logging policy that is set for the operation. For more information about the fields that are displayed in an API event record, see API event record field reference. For information about how to configure your logging preferences for API events, see activity-log policy and Including elements in your assembly.

- **Catalogs, spaces, and analytics**
  Understand how syndication and the use of catalogs and spaces affects the use of the features in API Connect analytics.
- **Accessing analytics**
  Analytics data is displayed in dashboards in the API Manager UI. Dashboards are scoped at the provider organization, catalog, and space level. To access analytics data at the cloud scope, use the Cloud Manager UI. Analytics data is also accessible by using the Toolkit CLI and REST API.

Catalogs, spaces, and analytics

Understand how syndication and the use of catalogs and spaces affects the use of the features in API Connect analytics.

The data for analytics is collated from API events that are logged when API operations are invoked. Analytics data for an API is scoped to the catalog or space where that API resides, and is only included in search results (and thus, visualizations and dashboards) for the owning catalog or space.

Access to the analytics data, and to the analytics functions in the API Manager user interface, can be managed through the use of catalogs and spaces, and the roles and permissions that are assigned to the users (or *members*) of the provider organization.

The cloud admin user can view aggregated analytics data for all catalogs, spaces, in all provider organizations, in the admin UI.

Catalogs, spaces, and permissions

Catalogs act as deployment targets through which APIs (in their containing plans and products) are staged and published to consumer organizations. API Manager users in the provider organization can be assigned the following access to the analytics component for a catalog:

- A role that has the Analytics > View permission for a catalog: These users can view the analytics data generated for the APIs in the catalog within *dashboards*, export dashboard data in its raw format or as event records, and apply filters to the data shown within the dashboards.
- A role that has the Analytics > Manage permission for a catalog: This is a legacy permission and has the same permissions as the View permission.

The IBM® API Connect syndication feature provides a way for you to partition a catalog into multiple deployment targets (or *spaces*) through which separate groupings of APIs (in their containing plans and products) can be staged and published. Each space can be allocated to a separate group of users who need to manage their products independently, and the analytics data in each space is scoped to those products only. API Manager users in the provider organization can be assigned the following access to the analytics component for a space:

- A role that has the Analytics > View permission for a space: These users can view the analytics data generated for the APIs in the space within dashboards, export dashboard data in its raw format, and apply filters to the data shown within the dashboards.
- A role that has the Analytics > Manage permission for a catalog: This is a legacy permission and has the same permissions as the View permission.

For more information about assigning catalog or space permissions to a role, see Managing Catalog membership and Managing Space membership.

For more information about the default API Manager roles and permissions, see API Connect user roles.

Accessing analytics

Analytics data is displayed in dashboards in the API Manager UI. Dashboards are scoped at the provider organization, catalog, and space level. To access analytics data at the cloud scope, use the Cloud Manager UI. Analytics data is also accessible by using the Toolkit CLI and REST API.

The ability to access analytics data at a catalog or space level depends on your assigned roles and permissions. For more information about catalogs and spaces, see Working with Catalogs and Using syndication in API Connect.

To access analytics data from the API Manager UI, select the Analytics tab when viewing a catalog. For more information about the analytics data viewable from the API Manager UI, see Accessing analytics from the API Manager UI.
Restriction: If the visualizations in your dashboards are empty or display no recent data, a possible reason might be that access to analytics data within API Connect is deactivated. Contact your administrator for confirmation.
To export your analytics data, see Export your analytics data.

To use the toolkit CLI or REST API to access your analytics data, see:

- Analytics CLI.
- Analytics REST API.

Configuring API governance in the API Manager

How to add custom API governance rulesets to your API development process, to validate and enforce organizational governance policies and best practices in your provider organization.

Before you begin

Before you can create API governance rulesets, the API governance optional add-on must be enabled on your management subsystem by your system administrator. See Enabling API governance on Kubernetes, and Enabling API governance on VMware for more information. If API governance is enabled in your deployment, the API governance resource is displayed on the Resources page in the API Manager.

One of the following roles is required to be able to configure API governance:

- Organization Administrator
- Owner
- Custom role with the **`Settings: Manage`** permission.

About this task

API governance is an optional add-on to IBM® API Connect that can be used to validate and enforce organizational governance policies and best practices in your API development process. You configure API governance by creating one or more custom rulesets that contain a collection of rules that can then be used to check Swagger, OpenAPI, and AsyncAPI documents.

API governance contains the following types of rulesets:

- Provider organization rulesets - these are custom rulesets that contain the rules that are created in, and are specific to, your provider organization.
- Global rulesets - these are pre-configured IBM and Spectral rulesets that contain the rules that are shared with your provider organization, and cannot be edited. Note that the Spectral ruleset names are prefixed by spectral-, and that their version matches the version of that ruleset that's available in Spectral.

API governance in IBM API Connect is based on the open-source Spectral linter; for more information about Spectral, see https://docs.stoplight.io/docs/spectral/674b27b261c3c-overview.

Note:

- API governance in IBM API Connect only supports the creation of custom rulesets that contain rules that use the built-in Spectral core functions, as defined in https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions. The use of custom functions, for example rules that use functions that you have created yourself in JavaScript files, is not supported.
- Some of the Spectral rules within the Global rulesets contain the property **`recommended: false`**, which means that those rules are ignored during validation. However, if you create a new ruleset from one of these rulesets by using the Save as new ruleset option, the **`recommended`** property isn't transferred to the new ruleset. Therefore all of the rules will be used in the validation, unless you delete those rules from the ruleset. The Spectral ruleset names are prefixed by spectral-.
- You can also configure API governance rulesets in the Cloud Manager. For more information, see Configuring API governance in the Cloud Manager.

The following methods can be used to configure API governance rulesets:

- Create a new ruleset by using the UI
- Import a ruleset
- Create a ruleset by using the CLI

Procedure

- To create a new ruleset by using the API Manager or API Designer UI, complete the following steps.
  1. In the API Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. In the Provider organization rulesets section, click Add > Create from scratch.
     The Create provider organization ruleset wizard opens.
  4. Provide the following Ruleset info details.

Property label	Required	Description	Data type
Title	Yes	A short descriptive name for the ruleset. This name is displayed in the list of global rulesets.	String
Name	Yes	An auto-generated, sluggified, string that is based on the title. It is not editable.	String
Version	Yes	The version number defaults to 1.0.0, but can be edited. You can have rulesets with the same name, but with different version numbers. The version number must be in the format **`major.minor.patch`**.	String
Description	No	An optional description of the ruleset.	String

5. Click Next, and then add an initial rule to your ruleset.
6. Provide the following Rule info details.

Property label	Required	Description	Data type
Title	Yes	A short descriptive name for the rule.	String
Name	Yes	A short machine name for the rule. Can contain only alpha-numeric characters, underscores (_), and hyphens (-).	String
Description	Yes	A description of the rule.	String
Message	No	Provide a message to help users understand what the goal of the rule is. If no message is provided, the value of the Description property is used for the message output of the validation scan.	String
Severity	Yes	Select a severity level from the following options: • error • warn • info • hint • off error is selected by default.	String

7. In the Given section, specify one or more values for the parts of the API document to target. To add more values, click Add. Values must be written in JSONPath.
8. In the Then section, specify the function type and options that will be used to evaluate the target values in the API document. Write this section in YAML syntax.
   In the following example, for a Given value of **$** (which means root level of the document), the Then section uses the Spectral core **truthy** function to check that the **info.contact** property value is not **false**, **""**, **0**, **null**, or **undefined**.

```
- field: info.contact
  function: truthy
```

   The Then section can also contain a list of functions to be applied to the Given part of the document. In the following example, for a Given value of **$.info.version** (which refers to the **version** property inside the **info** property at the root level of the document), the Then section uses both the **truthy** function, and the Spectral core **pattern** function that checks that the **info.value** matches the regex expression that is given by **functionOptions.match**.

```
- function: truthy
- function: pattern
  functionOptions:
    match: "^[0-9]+.[0-9]+.[0-9]+(-[a-z0-9+.-]+)?"
```

   For a complete list of the Spectral core functions that you can use in your custom rules, along with examples of these functions, see https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions.
9. Click Create to create your draft ruleset.
   Your draft ruleset opens in design form view. Note that you can switch to the OpenAPI YAML source view by clicking the Source icon </>. To return to the design form view, click the Form icon.
10. You now have the following options:
    - You can continue to edit your ruleset. For example, you can add more rules by clicking the Create rule icon alongside Rules in the navigation pane. Remember to click Save when you finish editing.
    - If you don't want to edit your ruleset further now, click API governance in the breadcrumb trail to return to the API governance overview page.

- To import an existing ruleset, complete the following steps. Remember that the rules in your imported ruleset can contain only built-in Spectral core functions, as defined in https://docs.stoplight.io/docs/spectral/cb95cf0d26b83-core-functions.
  1. In the API Manager, click Resources.
  2. In the Resources navigation menu, select API governance.
  3. In the Provider organization rulesets section, click Add > Import.
     The Import ruleset wizard opens.
  4. Drag a file into the Upload file box, or click in the Upload file box to select a file to upload. The maximum file size is 100 kb, and the supported file types are YAML, YML, and JSON.
  5. Click Next.
  6. Edit the Ruleset info details as required, and click Import.
     Your draft ruleset opens in design form view. Note that you can switch to the OpenAPI YAML source view by clicking the Source icon </>. To return to the design form view, click the Form icon.
  7. You now have the following options:
     - You can continue to edit your ruleset. For example, you can add more rules by clicking the Create rule icon alongside Rules in the navigation pane. Remember to click Save when you finish editing.
     - If you don't want to edit your ruleset further now, click API governance in the breadcrumb trail to return to the API governance overview page.
- To create a ruleset by using the toolkit CLI, complete the following steps.
  1. Create a JSON file containing the new ruleset.
     The following example shows the contents of a Spectral Ruleset file for API Manager:

```
{
  "name": "custom-ruleset",
  "title": "custom-ruleset",
  "description": "Info validation",
  "ruleset_type": "custom",
  "ruleset_version": "1.0.0",
  "rules": [
    {
      "name": "contact",
      "severity": "error",
      "formats": [
        "oas2",
        "oas3"
      ],
      "given": [
        "$"
```

```
      ],
      "description": "The info object must contain either a contact email or contact url",
      "then": [{
        "field": "info.contact",
        "function": "truthy"
      }]
    },
    {
      "name": "description",
      "severity": "error",
      "formats": [
        "oas2",
        "oas3"
      ],
      "given": [
        "$"
      ],
      "description": "The info object must contain a description",
      "then": [{
        "field": "info.description",
        "function": "truthy"
      }]
    }]
}
```

2. Log in to the toolkit as an the owner or an admin of the provider organization that owns the new ruleset.
   For example:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

3. Run the `rulesets:create` toolkit command in `governance` mode:

   ```
   apic -m governance rulesets:create --org <target_org> --server <server_url> RULESET_FILE
   ```

   where:
   - `<target_org>` is the name of the organization that owns the new ruleset.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `RULESET_FILE` is the path and filename of the JSON file that contains the ruleset.

Results

Your draft ruleset is now listed in the table of Provider organization rulesets.

What to do next

You can validate the ruleset against an API by clicking Validate, selecting the rules that you want to validate, and one or more APIs, then clicking Validate. The results of the validation are displayed in a scorecard.

When you finish editing your ruleset, you can publish it to your provider organization. Click the options menu icon ⋮ either next to the ruleset that you want to publish, or from within the ruleset viewer. Select Publish, and then click Publish again to confirm. The status of your ruleset changes from Draft to Published, and can now be used by API developers to validate their APIs. For more information, see Validating an API document by using API governance.

Note:

- After a ruleset is published, the ruleset information and rules can no longer be edited. If you need to update this information, you must create a new version by clicking the options menu icon ⋮ either next to the ruleset that you want to edit, or from within the ruleset viewer, and selecting Save as new ruleset.

- **The API governance ruleset lifecycle in the API Manager**
  When you manage your custom API governance rulesets in the API Manager UI, you move them through a series of lifecycle states. From initially creating a draft ruleset, through to publishing to make the ruleset available to your API developers, to creating new versions, and to eventual archiving.
- **Updating rulesets and rules with the toolkit CLI**
  Use the API Connect toolkit CLI to update a ruleset or individual rules.

The API governance ruleset lifecycle in the API Manager

When you manage your custom API governance rulesets in the API Manager UI, you move them through a series of lifecycle states. From initially creating a draft ruleset, through to publishing to make the ruleset available to your API developers, to creating new versions, and to eventual archiving.

The following sections describe the various lifecycle states for a custom API governance ruleset.

Note:

- Approval is not required for any of the lifecycle state transitions.
- Global rulesets are pre-configured rulesets that contain the rules that are shared with your provider organization, and cannot be edited. These rulesets are always in the Published state.

Draft

The draft state for a ruleset is when the ruleset is not published, and isn't visible to your provider organization. The ruleset can continue to be edited, and its Status is shown as Draft.

Published

When you publish a ruleset, it is made visible to your provider organization, and cannot be changed. The ruleset Status is shown as Published. Any changes to the ruleset require a new version to be created by using the Save as new ruleset option.

Archived

When you archive a ruleset, it becomes a ready-only, fixed version of the ruleset that cannot be used for validation by your provider organization. The ruleset Status is shown as Archived.
Note: If you cannot see your Archived ruleset, ensure that the Edit filters option on the ruleset table is set to include Archived.
You can restore archived rulesets by using the Restore option, after which the Status returns to Published.

Also, you can delete Archived rulesets by using the Delete option.

Updating rulesets and rules with the toolkit CLI

Use the API Connect toolkit CLI to update a ruleset or individual rules.

Updating a ruleset

1. Create a JSON file that defines the updated ruleset.
   The following example shows an updated ruleset:

   ```
   {
       "title": "custom-ruleset-2",
       "description": "short ruleset updated",
       "ruleset_type": "custom"
   }
   ```

   where:
   - `title` is updated to a new title for the ruleset.
   - `description` is updated to a new description for the ruleset.
   - `ruleset_type` is updated to be custom for the ruleset.
2. Log in to the toolkit as an Admin or as the owner of the provider organization that owns the ruleset.
3. Run the `rulesets:update` toolkit command in `governance` mode:

   ```
   apic -m governance rulesets:update --org <target_org> --server <server_url> RULESET RULESET_FILE
   ```

   where:
   - `<target_org>` is the name of the organization that owns the ruleset.
   - `<server_url>` is the management server URL that you use for logging in to the toolkit.
   - `RULESET_FILE` is the path and filename of the JSON file that contains the updated ruleset.
   - `RULESET` is the name or the unique ID of the updated ruleset.
     To determine the ID of a ruleset, run the following command:

     ```
     apic -m governance rulesets:list --org <target_org> --server <server_url>
     ```

Updating a rule

1. Create a JSON file that defines the updated rule.
   The following example shows an updated rule:

   ```
   {
     "description": "Updated the rule",
     "message": "Schema of type array must specify maxItems and minItems. {{path}} {{error}}",
     "formats": [
       "oas3"
     ],
     "severity": "warn",
     "recommended": true,
     "given": [
       "$..[?(@.type==\"array\")]"
   ```

```
  ]
}
```

where:

- `description` is updated to a new title for the rule
- `message` is updated to a new message for the rule
- `formats` array is replaced to have only the format oas3 for the rule
- `severity` is updated to the warn value for the rule
- `recommended` is updated to have a new value for the rule
- `given` array is replaced to have new values for the rule

2. Log in to the toolkit as an Admin or as the owner of the provider organization that owns the rule.
3. Run the `rules:update` toolkit command in `governance` mode.
   There are two ways to specify the updated rule in the command, depending on whether you provide the rule name and version, or the rule ID:

   Specify the rule name and version in the update command

   ```
   apic -m governance rules:update --org <target_org> --server <server_url> --ruleset <ruleset_name or ruleset_id>
   NAME:VERSION RULE_FILE
   ```

   where:

   - *`<target_org>`* is the name of the organization that owns the rule.
   - *`<server_url>`* is the management server URL that you use for logging in to the toolkit.
   - *`<ruleset_name or ruleset_id>`* is the name or the unique ID of the ruleset to which the updated rule belongs.
   - `NAME` is the name of the updated rule.
   - `VERSION` is the version of the updated rule; increase the version when you update the rule, using the `major.minor.patch` format as in "1.0.1".
   - `RULE_FILE` is the path and filename of the JSON file that contains the updated rule.

   Specify the rule ID in the update command

   Instead of providing the rule's name and version, you can provide its unique ID instead by running the following command:

   ```
   apic -m governance rules:update --org <target_org> --server <server_url> --ruleset <ruleset_name or ruleset_id> --id
   <rule_id> RULE_FILE
   ```

   where:

   - *`<target_org>`* is the name of the organization that owns the rule.
   - *`<server_url>`* is the management server URL that you specified when you logged in to the toolkit.
   - *`<ruleset_name or ruleset_id>`* is the name or the unique ID of the ruleset to which the update rule belongs.
   - `RULE_FILE` is the path and filename of the JSON file that contains the updated rule.
   - *`<rule_id>`* is the value of the updated rule's unique ID.
     To determine the ID of a rule, run the following command:

     ```
     apic -m governance rules:list --org <target_org> --server <server_url> --ruleset <ruleset_name> RULE_NAME
     ```

Administering user access

If you have permission to administer users in IBM® API Connect, you can add and delete users. After users are removed from an organization through deletion, the user account remains in API Manager.

About this task

You can also authorize users to perform different roles within your organization. To make any changes to the users that can access your organizations, use the followings tasks:

- **Setting the expiration for invitations**
  Set a time-out for invitations to expire.
- **Configuring notifications**
  Specify the sender name and email address to use for email notifications. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.
- **Viewing your user information**
  You can view the users that can access your IBM API Connect organization.
- **Adding provider organization users and assigning roles**
  If you have the permissions that are required to edit users in API Connect, you can add users to your provider organization, remove users, assign roles and perform other user administration tasks.
- **Creating custom roles**
  If you have permission to edit roles in IBM API Connect, you can create custom roles, and assign permissions, in a provider organization. You can create as many custom roles as you want.
- **Removing a user from a provider organization**
  If you have permission to edit users, you can remove a user from a provider organization.
- **Configuring LDAP group mappings on API Manager user roles**
  As an API Manager administrator, you can configure LDAP group mapping on API Connect roles in a provider organization by using the developer toolkit CLI.

Setting the expiration for invitations

Set a time-out for invitations to expire.

About this task

Invitations contain a link, which expires after the specified the timeout.

Procedure

1. In API Manager, click Settings.
2. In the Settings navigation list, click Onboarding, then click Edit.
3. To specify a timeout value, select or type an integer value in the Number field and then choose a unit of time (Seconds, Minutes, or Hours) in the Unit field.
4. Click Save.

Configuring notifications

Specify the sender name and email address to use for email notifications. You can preview the templates and customize the text if required. You can also style the notifications to match your business theme.

About this task

The API Manager emails are sent automatically when certain system events occur. The email templates are organized by scope, as explained in Table 1.

Table 1. Notification scopes

Scope	Events that trigger the notification
catalog	Activities related to Catalogs; such as application life cycle events, invitations to catalogs, product approvals, and subscription approvals.
consumer	Activities to consumer applications and subscription requests in the Developer Portal, invitations to Consumer organizations, and password reset requests for Developer Portal accounts.
provider	Invitations.
space	Invitations.

Email notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template being sent, the search can start at the Space level, and then go through the Catalog, Provider Organization, and the Cloud Manager levels. The sender details that are used will be the first set of configured details found during this hierarchical search. To view which search path API Connect takes when looking for the sender details for each notification template, see Configuring sender details for email notifications.

One of the following roles is required to configure email notifications:

- Administrator
- Owner
- A custom role with the `Settings:Manage` permission

Note: You can also customize notifications by using the developer toolkit CLI, or by using the API Connect REST APIs. See API development and management commands, or API Connect REST APIs, for more information.

Procedure

1. In API Manager, click Settings.
2. In the Settings navigation list, click Notifications, then click Edit.
3. Specify a sender by entering a Name and Email address, then click Save.
   To configure sender details at the Space or Catalog levels, you must use the developer toolkit CLI. For more information, see Configuring sender details for email notifications.
4. Optional: To enable template customization, move the Customize notification templates slider to the On position.
   Note: You can reload all the default notifications, as configured by a cloud administrator, by disabling and re-enabling template customization. However, doing so will overwrite any customizations that have been made. For more information on customizing notifications as a cloud administrator, see Customizing email notification templates.
5. To preview the text for a template, select Preview from the options menu alongside the template name.
6. To edit the text for the template, either click the template name, or select Edit in the options menu alongside the template name.
   When a template is opened, an attempt is made to get the language from the browser setting, but if this isn't possible the template defaults to English.

   The text includes variables; for example, `{{catalog}}`. The notification text is based on Handlebars syntax. Most variables are enclosed in double curly braces `{{ }}`, but can be enclosed in triple curly braces `{{{ }}}` to disable HTML escaping, when the variable is a URL link for example. For more information on Handlebars, see https://handlebarsjs.com/.

   To obtain the complete list of variables that are available for a particular notification template, complete the following steps:
   a. Log in to the management server from the command line as a member of a provider organization; for details, see Logging in to a management server. You can use the same management server URL, user name, and password in the login command that you use to log in to the API Manager user interface.
   b. Enter the following command:

      ```
      apic notification-templates:get template_name --server mgmt_endpoint_url --scope org --org provider_organization --subcollection template_scope --fields variables --output -
      ```

      where:
      - *template_name* is the name of the required notification template, as displayed in the Template column in the user interface.
      - *template_scope* is the scope name displayed in the Scope column alongside that template.
      - *provider_organization* is the value of the `name` field for your provider organization.

      For example:

      ```
      apic notification-templates:get member-invitation --server https://myserver.com --scope org --org myorg --subcollection catalog --fields variables --output -
      ```

      The variables that are available for the template are displayed, for example:

```
variables:
  - org
  - catalog
  - activationLink
  - expiresAt
  - originator
  - originatorFirstName
  - originatorLastName
  - originatorEmail
  - username
  - email
  - firstName
  - lastName
```

The `--output -` parameter causes the command output to be written to the command line. You can specify `--output` ***`filepath`*** to have the output written to a .yaml file at the specified location, or omit it altogether to have a file written to the current folder.

7. To view and edit the notification template in a different language, select one of the following supported languages from the View template in drop-down list:
   - Chinese (Simplified)
   - Chinese (Traditional)
   - Czech
   - Dutch
   - English (US English)
   - French
   - German
   - Italian
   - Japanese
   - Korean
   - Polish
   - Portuguese
   - Russian
   - Spanish
   - Turkish
8. Edit the Subject as required.
9. Select the Content type that you want to use for the template, from HTML, PlainText, or Both. The default content type is PlainText.
   An edit window for the selected content type is displayed, or both edit windows are displayed if Both is selected.
10. Edit the body of the template as required.
    For HTML content, only the tags and their attributes that are shown in the following table are allowed.

Table 2. List of allowed HTML tags and their attributes

HTML tag	Attribute
`<a>`	`"class", "href", "hreflang", "style"`
`<em>`	`"class", "style"`
`<strong>`	`"class", "style"`
`<cite>`	`"class", "style"`
`<blockquote>`	`"class", "cite", "style"`
`<code>`	`"class", "style"`
`<ul>`	`"class", "type", "style"`
`<ol>`	`"class", "start", "type", "style"`
`<li>`	`"class", "style"`
`<dl>`	`"class", "style"`
`<dt>`	`"class", "style"`
`<dd>`	`"class", "style"`
`<h1>`	`"class", "id", "style"`
`<h2>`	`"class", "id", "style"`
`<h3>`	`"class", "id", "style"`
`<h4>`	`"class", "id", "style"`
`<h5>`	`"class", "id", "style"`
`<h6>`	`"class", "id", "style"`
`<p>`	`"class", "style"`
`<div>`	`"class", "style"`
` `	`"class", "style"`
`<span>`	`"class", "style"`
`<img>`	`"class", "src", "alt", "data-entity-type", "data-entity-uuid", "data-align", "data-caption", "width", "height", "style"`
`<table>`	`"class", "id", "style"`
`<tr>`	`"class", "id", "style"`
`<td>`	`"class", "id", "style"`

If an HTML tag that is not allowed is used in a notification, the tag and its contents are displayed in the email as plain text.
Images can be used by adding a `<img src="https://path/to/image.png"/>` tag in the template. The `src` attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

11. Click Save when done.
    Note: Edits made to a template are saved only for the specific language version that is edited.

- **Configuring sender details for email notifications**
  Understand how the hierarchical search works for the email notification sender details, and how you can configure sender details at the Space and Catalog level.

Configuring sender details for email notifications

Understand how the hierarchical search works for the email notification sender details, and how you can configure sender details at the Space and Catalog level.

Before you begin

One of the following roles is required to configure sender details for email notifications:

- Administrator
- Owner
- A custom role with the `Cloud settings:Manage` or `Settings:Manage` permission

For more information about the notification templates, their scope, and how to customize them, see Configuring notifications at the Provider Organization level, and Customizing email notification templates at the Cloud Manager level.

About this task

Email notifications are sent automatically by API Connect when certain system events occur. These notifications contain a sender name and address that's based on a hierarchical search of the configured sender details in API Connect. Depending on the email template that's being sent, the search can start at the Space level, the Catalog level, the Provider Organization level, or the Cloud level. The sender details that are used are the first set of configured details that are found during the hierarchical search.
The following table shows the starting point for the sender details hierarchical search for each email template, ordered by the scope of the template, where:

- Cloud: emails are sent from the details that are configured in the Settings > Notifications section of the Cloud Manager.
- Org: the search for sender details starts at the Provider Organization level; if not available, the Cloud details are used.
- Catalog: the search for sender details starts at the Catalog level; if not available, the Org details are used, and if they're not available, the Cloud details are used.
- Space: the search for sender details starts at the Space level; if not available, the Catalog details are used, if they're not available, the Org details are used, and if they're not available, the Cloud details are used.
- Space or Catalog: in these cases the search starts at the Space level, or at the Catalog level, depending on the user that made the request, as shown in the following list:
  - When the product lifecycle change is started by a user in a Space, the search for sender details starts at the Space level.
  - When the product lifecycle change is started by a user in a Catalog, the search for sender details starts at the Catalog level.
  - When the product lifecycle change is approved by a user in a Space, the search for sender details starts at the Space level.
  - When the product lifecycle change is approved by a user in a Catalog, the search for sender details starts at the Catalog level.

Table 1. Starting point for the sender details search for each email template

Scope	Template name	Email subject	Sender details
admin	mail-server-test-connection	Test message from IBM® API Connect.	Cloud
admin	member-invitation	Invitation to an admin organization in IBM API Connect.	Cloud
admin	password-changed	Password changed for your user account in IBM API Connect.	Cloud
admin	password-reset	Password reset request for your user account in IBM API Connect.	Cloud
provider	invitation	Invitation to create an API provider organization in IBM API Connect.	Cloud
provider	member-invitation	Invitation to an API provider organization in IBM API Connect.	Org
catalog	app-lifecycle-cancelled	Request withdrawn to *{{action}}* app *{{appName}}*.	Catalog
catalog	app-lifecycle-request	Request for approval to *{{action}}* app *{{appName}}*.	Catalog
catalog	invitation	Invitation to create an API catalog in IBM API Connect.	Org
catalog	member-invitation	Invitation to an API catalog in IBM API Connect.	Catalog
catalog	task-consumer-onboard-request	Consumer onboarding request in the *{{catalog}}* developer portal.	Catalog
catalog	task-product-lifecycle-approved	Request approved to *{{action}}* API product in catalog *{{catalog}}*.	Space or Catalog
catalog	task-product-lifecycle-cancelled	Request withdrawn to *{{action}}* an API product in the *{{catalog}}* catalog.	Space or Catalog
catalog	task-product-lifecycle-denied	Request denied to *{{action}}* API product in catalog *{{catalog}}*.	Space or Catalog
catalog	task-product-lifecycle-pending	Request received to *{{action}}* API product in catalog *{{catalog}}*.	Space or Catalog
catalog	task-product-lifecycle-request	Request for approval to *{{action}}* an API product in the *{{catalog}}* catalog.	Space or Catalog
catalog	task-subscription-cancelled	Request withdrawn to subscribe to an API product in the *{{catalog}}* catalog.	Catalog
catalog	task-subscription-request	Request for approval to subscribe to an API product in the *{{catalog}}* catalog.	Catalog
space	invitation	Invitation to create an API space in IBM API Connect.	Catalog
space	member-invitation	Invitation to an API space in IBM API Connect.	Space
consumer	account-approved	*{{catalogTitle}}* developer portal account registration is approved.	Catalog
consumer	account-denied	*{{catalogTitle}}* developer portal account registration is denied.	Catalog
consumer	account-pending-approval	*{{catalogTitle}}* developer portal account registration pending approval.	Catalog
consumer	app-lifecycle-approved	Request approved to *{{action}}* app *{{appName}}*.	Catalog
consumer	app-lifecycle-denied	Request denied to *{{action}}* app *{{appName}}*.	Catalog
consumer	app-lifecycle-pending	Request received to *{{action}}* app *{{appName}}*.	Catalog
consumer	app-reinstated	*{{appName}}* app reinstated in the *{{consumerOrg}}* developer portal.	Catalog
consumer	app-suspended	*{{appName}}* app suspended in the *{{consumerOrg}}* developer portal.	Catalog
consumer	invitation	Invitation to create an API consumer organization in the *{{catalog}}* developer portal.	Catalog
consumer	member-invitation	Invitation to an API consumer organization in the *{{catalog}}* developer portal.	Catalog
consumer	password-changed	Password changed for your user account in *{{portalTitle}}* developer portal.	Catalog
consumer	password-reset	Password reset request for your *{{catalog}}* developer portal account.	Catalog
consumer	sign-up	*{{catalogTitle}}* developer portal account registration.	Catalog
consumer	task-subscription-approved	API subscription request approved for app *{{appName}}*.	Catalog
consumer	task-subscription-denied	API subscription request denied for app *{{appName}}*.	Catalog
consumer	task-subscription-pending	API subscription request received for app *{{appName}}*.	Catalog

The sender address details can be configured at four levels: Cloud, Provider Organization, Catalog, and Space, and you can set a different sender address for each Provider Organization, Catalog, or Space, as required. The sender address for the Cloud can be configured in the Cloud Manager UI; see Setting up notifications for details. The

sender address for the Provider Organization can be configured in the API Manager UI; see Configuring notifications for details. However, the sender address for Catalogs and Spaces can be configured only by using the toolkit CLI, as explained in the following steps.

Procedure

1. Log in to the management server from the command line as a member of a provider organization; for more information, see Logging in to a management server. You can use the same management server URL, username, and password in the login command that you use to log in to the API Manager user interface.
2. To update the sender address details for a Space, run the following commands.
   a. Run the **apic space-settings:get** command to read the current Space settings and output them to a json file. For example:

   ```
   apic space-settings:get --server my_mgt_server --org my_provider_org --catalog my_catalog --space my_space --format
   json
   ```

   Where:
   - ***my_mgt_server*** is your management server endpoint or URL.
   - ***my_provider_org*** is the name of your Provider Organization.
   - ***my_catalog*** is the name of your Catalog.
   - ***my_space*** is the name of your Space.

   b. Make a copy of the output file that you can then use to update the settings, for example:

   ```
   cp space-setting.json space-setting-update.json
   ```

   c. Edit the copied settings file to update the **email_sender** block. For example:

   ```
   vi space-setting-update.json
   ...
     "email_sender": {
       "custom": true,
       "name": "NO REPLY: space sender name",
       "address": "space_sender_email@nomail.com"
     },
   ...
   ```

   Where:
   - **custom** must be changed from the default setting of **false** to **true**.
   - the **name** line must be added to the file, and the ***space sender name*** details completed.
   - the **address** line must be added to the file, and the ***space_sender_email@nomail.com*** address details completed.

   Note: Ignore all of the other configuration details that are contained in this settings file.

   d. Run the **space-settings:update** command to update the Space settings with the new sender address details. For example:

   ```
   apic space-settings:update --server my_mgt_server --org my_provider_org --catalog my_catalog --space my_space space-
   setting-update.json
   ```

   Where ***space-setting-update.json*** is the name of your updated settings file.
3. To update the sender address details for a Catalog, run the following commands.
   a. Run the **apic catalog-settings:get** command to read the current Catalog settings and output them to a json file. For example:

   ```
   apic catalog-settings:get --server my_mgt_server --org my_provider_org --catalog my_catalog --format json
   ```

   Where:
   - ***my_mgt_server*** is your management server endpoint or URL.
   - ***my_provider_org*** is the name of your Provider Organization.
   - ***my_catalog*** is the name of your Catalog.

   b. Make a copy of the output file that you can then use to update the settings, for example:

   ```
   cp catalog-setting.json catalog-setting-update.json
   ```

   c. Edit the copied settings file to update the **email_sender** block. For example:

   ```
   vi catalog-setting-update.json
   ...
     "email_sender": {
       "custom": true,
       "name": "NO REPLY: catalog sender name",
       "address": "catalog_sender_email@nomail.com"
     },
   ...
   ```

   Where:
   - **custom** must be changed from the default setting of **false** to **true**.
   - **name** line must be added to the file, and the ***catalog sender name*** details completed.
   - **address** line must be added to the file, and the ***catalog_sender_email@nomail.com*** address details completed.

   Note: Ignore all of the other configuration details that are contained in this settings file.

   d. Run the **catalog-settings:update** command to update the Catalog settings with the new sender address details. For example:

   ```
   apic catalog-settings:update --server my_mgt_server --org my_provider_org --catalog my_catalog catalog-setting-
   update.json
   ```

   Where ***catalog-setting-update.json*** is the name of your updated settings file.

Results

You now understand how the hierarchical search works for the sender details for each email template, and can update the sender details at the Space and Catalog level as required.

Viewing your user information

You can view the users that can access your IBM® API Connect organization.

Procedure

To view your organization user information, complete the following steps:

In the navigation pane of the API Manager user interface, click Members.

What to do next

You can modify your user accounts; see Adding provider organization users and assigning roles. You can create custom roles; see Creating custom roles.

Adding provider organization users and assigning roles

If you have the permissions that are required to edit users in API Connect, you can add users to your provider organization, remove users, assign roles and perform other user administration tasks.

About this task

There are two ways to add a user:

- Invite member - send an invitation email with an activation link that enables the user to complete the addition operation.
- Add member - specify the user and add them as a member of the provider organization immediately.

Procedure

- To send an email invitation to a new user to register as a member of the provider organization, complete the following steps:
  1. In the navigation pane of the API Manager user interface, click Members.
  2. Click Add > Invite member.
  3. Enter the email address of the user.
  4. Select the roles that you want to assign to the user.
  5. Click Invite.
- To specify the user and add them as a member of the Admin organization immediately, complete the following steps:
  1. In the navigation pane of the API Manager user interface, click Members.
  2. Click Add > Add member.
  3. Select the required user registry.

     Note: For a user registry to be available in the selection list, one of the following conditions must be satisfied:
     - The visibility of the user registry is set to Public.
     - The visibility of the user registry is set to Custom, and the specified list of provider organizations includes the one to which you are adding the user.

     For more information, see Setting visibility for a user registry.

     The remaining procedure varies according to the type of the selected user registry, as follows:
     - Local User Registry
       - Select whether the user is an Existing user or a New User.
         - For an existing user, complete the following steps:
           - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
           - Select the roles that you want to assign to the user, then click Add. The user is added and their membership is immediately enabled. The specified user can log in to the API Manager user interface.
         - For a new user, complete the following steps:
           - Enter a unique user name for the new user.
           - Supply an email address, name details, and a password.
           - Select the roles that you want to assign to the user, then click Add. The user account is created, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface.
     - LDAP
       - Enter the name of a user that exists in the selected user registry.
       - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface with their LDAP user name.
     - Authentication URL and OIDC
       - Enter the name of an existing API Connect user that has previously been invited to register and has activated their account.
       - Select the roles that you want to assign to the user, then click Add. The user is added, and the user membership is immediately enabled. The specified user can log in to the API Manager user interface.

Results

The user is added to the list of provider organization members. If an email invitation was sent, the status is shown as `Pending` until the recipient of the email clicks the link in the email to complete the creation of their account, after which the status changes to `Enabled`.

What to do next

The new user can access the API Manager user interface. The user's authorization within API Manager is defined by the roles that are assigned to them.

Related tasks

- Creating custom roles

Creating custom roles

If you have permission to edit roles in IBM® API Connect, you can create custom roles, and assign permissions, in a provider organization. You can create as many custom roles as you want.

About this task

You create custom roles at the provider organization level; those roles are inherited by the Catalogs and Spaces in the provider organization. You can assign permissions at the provider organization, Catalog or Space level.

For a description of the permissions, and details of the default user roles and default permissions assigned to those roles, see API Connect user roles.

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

Procedure

To create a custom role, complete the following steps:

1. In the navigation pane of the API Manager user interface, click Settings
2. Click Roles, then click Add.
   The Create a Role page opens.
3. Enter a role Title and an optional Summary. A Name is entered automatically.
   Note: The value in the Name field is a single string that is used to identify the role in developer toolkit CLI commands. The Title is used for display.
   To view the CLI commands to manage roles, see apic roles.
4. Use the check boxes to assign permissions to the new role.
5. When you are finished, click Save.
6. To delete a role, click the options icon alongside the role that you want to delete, click Delete, then click Delete again to confirm the role deletion.
   Note:
   - You can create and delete a role only at the provider organization level. You cannot create or delete a role at the Catalog level. Nor can you create or delete a role at the Space level.
     You can, however, assign Catalog-specific permissions to the role; for details, see Creating and configuring Catalogs. You can also assign Space-specific permissions; for details, see Managing user access in a Space; for more information on Spaces, see Using syndication in API Connect.
   - If you assign Product-Drafts or Api-Drafts permissions to the role, these permissions are not inherited by the role in a Catalog or Space. These permissions apply to working with draft Products and APIs in a provider organization and are not relevant in a Catalog or Space.

Results

The custom role is created and assigned the permissions that you selected.

What to do next

Assign the custom role to a user.

Removing a user from a provider organization

If you have permission to edit users, you can remove a user from a provider organization.

About this task

After you have removed the user from your IBM® API Connect provider organization, their resources are deleted and the user cannot access any of the organization's artifacts.
Note: The user account of the deleted user remains in the associated API Connect user registry. The user can still log in to the API Manager user interface but only information links are available; there is no access to view or manage any resources.
If you later invite the same user again, they can re-activate their account. If you are using a Local User Registry, the user must re-activate their account by using the Sign In option, **not** by completing the registration form and using the Sign Up option; attempting to re-register will fail. For details on how to invite a user to join your organization, see Adding provider organization users and assigning roles.

Procedure

To remove a user from a provider organization complete the following steps:

1. In the navigation pane of the API Manager user interface, click Members.
2. Alongside the user that you want to delete, click the options icon , click Delete, the click Delete again to confirm the user deletion.
   Note: You cannot delete the owner of a provider organization.

Results

The user is removed from the provider organization, and the user account remains in API Manager.

Configuring LDAP group mappings on API Manager user roles

As an API Manager administrator, you can configure LDAP group mapping on API Connect roles in a provider organization by using the developer toolkit CLI.

Before you begin

You must have an LDAP user registry resource in the API Manager that has the **`external_group_mapping_enabled`** configuration set to **`true`**. See Using the CLI to create an organization-specific LDAP user registry for information.

One of the following roles is required to edit roles:

- Administrator
- Owner
- Custom role with the `Settings: manage` permissions

Note: The provider organization roles apply only to API Manager users. If you want to apply LDAP group mapping to Cloud Manager users, see Configuring LDAP group mapping on Cloud Manager user roles. LDAP group mapping cannot be applied to Developer Portal users.

About this task

You can map external LDAP groups to the API Connect preconfigured user roles (except for the Owner and Member roles), as well as to any custom user roles, to reflect your business needs.

Notes:

- After LDAP group mapping is enabled on a role, user onboarding always honors the group mappings.
- Once on-boarded, user membership in API Connect is valid throughout the login period (**`access_token_ttl`**), irrespective of any changes in the external LDAP registry. Membership is updated only on the next login, when the LDAP information is fetched and refreshed.
- One or more API Connect roles can be mapped to one or more LDAP groups, and one or more LDAP groups can be mapped to a role.
- When multiple LDAP groups are mapped to a single role, it means that a user from any one of the LDAP groups can logon to API Connect.
- If a user is removed from the external LDAP user registry, to ensure quick removal from API Connect you must also delete the user membership in API Connect.
- In API Manager you can map roles at the provider organization, Catalog and, if appropriate, Space level. However, you can only map roles at the child level of Catalog, and then the Catalog's child level of Space, if the parent level also has group mapping defined. Mapping at a child level overrides the mapping for the same role at the parent level. For example:
  If the Administrator Role at the provider organization level is mapped to the LDAP group of **`cn=APIC-Administrators,ou=ibmgroups,o=ibm.com`**, and the Administrator Role at a Catalog level is mapped to the LDAP group of **`cn=APIC-Developers,ou=ibmgroups,o=ibm.com`**, then any user that is added as a Catalog member must belong to the **`cn=APIC-Developers,ou=ibmgroups,o=ibm.com`** group. Note that provider organization level members that belong to the **`cn=APIC-Administrators,ou=ibmgroups,o=ibm.com`** group can still access the Catalog due to inheritance.

You can configure LDAP group mappings using one of the following methods:

- Using the UI to configure LDAP group mappings
- Using the CLI to configure LDAP group mappings

Using the UI to configure LDAP group mappings

Use the API Manager UI to configure LDAP group mappings.

Procedure

1. Login to the API Manager as a user with the appropriate permissions.
2. Click Settings > Roles > Add a new Role or edit existing role .
3. Select the following settings:
   - Enable external group mapping - select to enable external group mapping for this role.
   - LDAP user Registry - LDAP user registry list which has external group mapping enabled. User must select one of the LDAP user registry from this list.
   - LDAP group names - add one or more LDAP group names that you want to map to the user role.
   - User group prefix - add the prefix for the LDAP user group.
   - User group suffix - add the suffix for the LDAP user group.
4. Save your changes.

Results

The role is updated with the LDAP group mapping information. Users can now log on to the API Manager, and automatically be assigned the correct access permissions for their role.

Using the CLI to configure LDAP group mappings

Use the toolkit CLI to configure LDAP group mappings.

About this task

For detailed information about how to use the CLI, see Installing the toolkit, and Overview of the command-line tool.

Procedure

Perform the following steps to map external LDAP groups to API Manager user roles.

1. Log in to the management server CLI.
   Before you can update the role configuration, you must log in to your management server from the developer toolkit CLI as a member of a provider organization. Use the following command:

   ```
   apic login --server mgmt_endpoint_url --username user_id --password password --realm provider/identity_provider
   ```

   where ***mgmt_endpoint_url*** is the platform API endpoint URL.
   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
   For full details of the login command, see Logging in to a management server.

2. Run the following command to get the URL of the LDAP user registry resource in the API Manager that you want to map the user roles to:

   ```
   apic user-registries:get  ldap_user_registry --org organization_name --server mgmt_endpoint_url --output -
   ```

   where:
   - ***ldap_user_registry*** is the name or ID of your LDAP user registry resource.
   - ***organization_name*** is the value of the **name** property of your provider organization.
   - ***mgmt_endpoint_url*** is the platform API endpoint URL.

   This command outputs the configuration details of your LDAP user registry, and the **url:** is shown at the end of the list, for example:

   ```
   type: user_registry
   api_version: 2.0.0
   id: 35e75bad-1d89-4a65-a70f-xxxxxx
   name: ldap
   title: LDAP
   integration_url: >-
     https://server.com/api/cloud/integrations/user-registry/147b5fb1-e88e-41e3-90e9-xxxxxx
   registry_type: ldap
   user_managed: false
   user_registry_managed: false
   external_group_mapping_enabled: true
   ...
   url: >-
     https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
   ```

3. Create a role yaml file that contains the following LDAP group mapping configuration properties:

   ```
   external_group_mapping:
     user_registry_url: https://server.com/api/user-registries/3d58ce7e-16a8-493b-9684-xxxxxx/35e75bad-1d89-4a65-a70f-xxxxxx
     ldap_groups:
       - 'cn=apic-administrators,ou=ibmgroups,o=ibm.com'
       - 'cn=apic-developers,ou=ibmgroups,o=ibm.com'
     user_group_filter_prefix: (&(uniquemember=
     user_group_filter_suffix: )(objectClass=groupOfUniqueNames))
   ```

   Where:
   - **user_registry_url** is the URL of your LDAP user registry resource from Step #task_apim_role_mapping__ldap_url.
   - **ldap_groups** is a list of the LDAP group names that you want to map to the user role.
   - **user_group_filter_prefix** is the prefix for the LDAP user group.
   - **user_group_filter_suffix** is the suffix for the LDAP user group.

4. Run the following command to update the user role with the **external_group_mapping** configuration properties:

   ```
   apic roles:update role_name --scope org --org organization_name --server mgmt_endpoint_url mapping_properties_file
   ```

   Where:
   - ***role_name*** is the name of the user role that you want to add the LDAP group mapping to.
   - **--scope** is the organization level that you want the update to apply to. Valid values are:
     - **org** to apply the mapping at the provider organization level.
     - **catalog** to apply the mapping at the Catalog level.
     - **space** to apply the mapping at the Space level.
   - ***organization_name*** is the value of the **name** property of your provider organization.
   - ***mgmt_endpoint_url*** is the platform API endpoint URL.

- ***mapping_properties_file*** is the name of your mapping properties file from Step #task_apim_role_mapping__role_file, for example **role_mapping_file.yaml**.

If you prefer to enter the configuration properties interactively on the command line, you can substitute the ***mapping_properties_file*** for a terminating hyphen character **-**, and enter the information manually, followed by pressing **CTRL D** to terminate the input.

If you want to create a custom role that includes LDAP group mapping, you can include the **external_group_mapping** configuration section in the **role_file**, and then create the new role by using the **apic roles:create** command.

For more information about the **apic roles** commands, see **apic roles** in the CLI reference section.

Results

The role is updated with the LDAP group mapping information. Users can now log on to the API Manager, and automatically be assigned the correct access permissions for their role.

Changing your API Manager password

You can change your API Manager password in IBM® API Connect.

Procedure

To change your password, complete the following steps:

1. Log in to the API Manager user interface.
2. Click the User icon then click My Account.
3. In the Change password section, enter your current password and your new password, and confirm the new password.
   Passwords must satisfy the following requirements:
   - Cannot be re-used for at least 8 password cycles
   - Cannot contain more than 2 consecutive repeating characters
   - Must include at least 8 characters
   - Must include 1 or more characters from at least 2 of the following categories:
     - Lowercase letters
     - Uppercase letters
     - Numbers
     - Special characters – only the following special characters are allowed:

       ```
       !
       \"
       #
       $
       %
       &
       \
       (
       )
       *
       +
       ,
       -
       .
       /
       :
       ;
       <
       =
       >
       ?
       @
       [
       \\
       ]
       ^
       _
       `
       {
       |
       }
       ~
       ```
4. Click Change Password.

Results

Your password is changed.

Resolving login problems by increasing HTTP header size

You can resolve login problems for the API Manager UI by increasing the maximum HTTP client header size.

About this task

Login attempts to the user interface might fail with error **`502 (Bad Gateway)`**. This error can occur when logging in from a browser that has a large number of cookies. The error occurs because size of the login request exceeds the maximum HTTP client header size.

The maximum HTTP client header size is limited for security reasons. To workaround this issue, you can clear the browser cache and cookies, or open an incognito window from the browser, and then retry the login.

Alternatively, you can increase the maximum HTTP client header size. Use caution when increasing the maximum size because larger headers can raise a security risk to your system.

Procedure

1. **SSH** to your management system or appliance.
2. Enter the following commands:

```
kubectl get deployment -n namespace  | grep apim
kubectl edit deployment -n namespace management-apim
```

3. In the **env** section of the deployment configuration, change **--max-http-header-size=12000** to a larger value that works for your environment. For example:

```
 - name: NODE_OPTIONS
          value: --max-old-space-size=8094 --max-http-header-size=16000
```

4. Save the updated configuration.
5. Run `kubectl get pod -n namespace` a few times until the new `management-apim` pods are up and running.

API Gateway only

Reviewing a gateway's processing status

Review the processing status for the Gateway services that are enabled in your API Connect Catalogs and Spaces.

About this task

Sometimes you might want to check the status of an API event that is being processed by the gateway. For example, you might want to ensure that a Product was published before and is available on the gateway before you invoke one of its APIs. Or perhaps it appears that API events are being processed too slowly and you want to see if all of the events for a particular gateway service are affected. You can review a gateway's processing status to see the which events were sent for processing and which events are still waiting in the queue.

Notes:

- You can only obtain processing status information for the DataPower® API Gateway. This feature is not supported for the DataPower Gateway (v5 compatible) service.
- Processing information is collected for each Catalog. If you view information for a particular Space, the display shows the results for the entire Catalog instead.

Procedure

1. Open the Gateway services page:
   a. Log in to API Manager.
   b. In the navigation pane, click Manage.
   c. Select a Catalog.
   d. Click the Catalog settings tab.
   e. On the Settings page, click Gateway services.
   A list of all of the gateway services for the selected Catalog, including all of its Spaces, displays the gateway type and its endpoint URL. In addition, a status icon displays for each service name to provide a quick indication of that service's current status:
   - Normal: Between 0 and 10 events are waiting to be processed
   - Increased: Between 10 and 20 events are waiting to be processed
   - High: More than 20 events are waiting to be processed

   You can view details about a service's status to gain insight into possible issues.
2. On the Gateway services page, click ⋮ next to a service and select View status.
   The Gateway processing status page displays a list of the events that were sent to the gateway. For each event, you can see the following information:
   - Event type: The action performed by the event; for example, subscribing to a product, deleting an application, or creating a new consumer organization.
   - Category: The feature area affected by the event; for example, subscriptions, applications, or consumer organizations.
   - Time: The time when the event was processed by the gateway service.
   - Status: The current event's processing status; for example, Queued (not yet sent for processing), Sent (but not processed yet), or Processed (complete). The number of events in the Queued and Sent states determine the status level for the service.

   The display includes events that were not processed yet, as well as the last event processed. If a particular event is not listed as Sent or Queued, then that event was already processed.

Reference

Reference information for the API Manager component in IBM® API Connect.

- **API gateway response codes**
  When an API is called, different HTTP status codes are returned by the gateway to indicate whether the request was successfully completed.
- **Troubleshooting your billing configuration**
  Some issues can occur when you are setting up billing integration in IBM API Connect.

Related information

- IBM API Connect overview

API gateway response codes

When an API is called, different HTTP status codes are returned by the gateway to indicate whether the request was successfully completed.

The response codes used in IBM® API Connect correspond to the registered HTTP status codes that are typically generated to provide informational (1xx), successful (2xx), redirection (3xx), client error (4xx), or server error (5xx) responses, as described at https://tools.ietf.org/html/rfc7231#section-6 and Hypertext Transfer Protocol (HTTP) Status Code Registry.

In API Connect, successful responses vary depending on the API being called. Other response codes can also be generated, depending on the implementation of the assembly and the response from the external systems. The standard reasons listed for the registered HTTP status codes are considered adequate for most responses that are returned; these response codes and their causes are therefore not listed here.

In certain cases, a client or server error response code can be caused by a condition that is specific to API Connect. The following table contains a list of these error response codes and identifies possible causes for these codes being returned. For some error codes, multiple causes are possible.

Table 1. Error codes and their causes

Error Code	Cause
401 Unauthorized	• The required client identification has not been successfully provided. • User authentication failed or did not take place.
403 Forbidden	• The application is not registered with the plan that is used. • The application is not active. • User authentication failed because multiple client IDs provided.
404 Not Found	• Information for the provider organization or environment was not found. • The API URL was not found in the organization or environment.
405 Method Not Allowed	The API URL was found, but no operation was found that supports the requested HTTP verb.
406 Not Acceptable	The API cannot produce any responses that are supported by the application.
429 Too Many Requests	The rate limit has been exceeded for the plan or operation being used.
500 Internal Server Error	An error occurred while executing this request.
503 Service Unavailable	The status of an API was switched from online to offline, making the API unavailable across all Products in which it is contained. For more information, see Managing your Products in the API Manager UI and Managing API Products using the developer toolkit.

Troubleshooting your billing configuration

Some issues can occur when you are setting up billing integration in IBM® API Connect.

You can use the following links to browse the issues:

1. My customer payment is not completing with Stripe, but the Product is still being accessed.
2. My Product with billing is returning an error when I try to configure it with my Stripe account.
3. My customer previously had a Stripe subscription to a Product but can't access its billing details.

My customer payment is not completing with Stripe, but the Product is still being accessed.

When payment issues occur with the credit card payment provider, the credit card subscription is canceled, but the API Connect subscription is not canceled. You can disable an application until the payment issues are resolved by completing the following steps:

1. Navigate to the Settings page in the dashboard of your Stripe account.
2. In the Billing section navigate to the Subscriptions and emails page.
3. In the Manage failed payments section, change the Subscription status to If all retries for a payment fail, mark the subscription as unpaid.
4. Navigate to the Subscriptions page, and when the subscription has moved into the unpaid state open the subscription in the Stripe dashboard.
5. In the Metadata section, copy the subscription_url value.
6. Use the developer toolkit CLI, and the UUID from the subscription_url, to set the state of the application or subscription to disabled.

My Product with billing is returning an error when I try to configure it with my Stripe account.

The most common cause of this is that your Developer Portal cluster servers or your Management cluster servers cannot communicate with Stripe.
To resolve this, ensure that the Stripe API is enabled with HTTPS communication with your Management cluster servers and the Developer Portal on port 443. See Firewall requirements on Kubernetes and Firewall requirements on VMware for more information about this requirement.

Other causes might be issues with the subscription billing service, network stability, or the API keys might have been rolled and not updated in API Connect and the Developer Portal.

My customer previously had a Stripe subscription to a Product but can't access its billing details.

There is more than one possible cause and resolution for this situation.

- The credit card transaction might not be completing correctly. If this situation happens, the subscription might be canceled with Stripe, but the API Connect subscription is not canceled. For instructions about how to disable a subscription, see My customer payment is not completing with Stripe, but the Product is still being accessed.
- A recent change to a Product with billing might have triggered an action that was not completed correctly. Seemingly minor changes to the Product can start a number of actions that can overload a server. For example, when you replace a version of the Product with a new version, it starts an update to the Stripe server for every account that is subscribed to that Product. If the action fails to complete, the Plan's subscription status might not be reflected correctly with Stripe.
  You can identify this situation by viewing the Billing dashboard in the Resources section of the API Manager. The table displays the state of the job queue for the billing integration resource, and any issues with the job queue are displayed here. For more information about the job queue, and how to retry blocked jobs, see Billing integration resource job queues.

Authoring policies

With the on-premise offering, you can control specific processing features in the Gateway server of IBM® API Connect by creating user-defined policies.

IBM API Connect enables organizations to easily promote business services as APIs to internal and external developer communities. API developers can rapidly create, proxy, assemble, and secure APIs through the API Designer user interface (UI). The API Designer assembly feature allows API developers to create complex REST or SOAP API operations that transform data, perform multiple service calls, aggregate data, and apply policies.

A policy is a piece of configuration that controls a specific aspect of processing in the IBM DataPower® Gateway server during the handling of an API invocation at run time. IBM API Connect provides a number of different types of policies, but you can also create user-defined policies to provide more processing control.

When would you need a user-defined policy?

A user-defined policy enables you to control extra processing features in the Gateway server, such as security, or routing of requests. The user-defined policies feature is available only with the on-premise offering of IBM API Connect.

Create a user-defined policy in IBM API Connect when you need to augment the actions or activities that are performed by the API gateway. For example, you might want to perform the following operations:

- Implement your own proprietary logic for dynamic routing of requests.
- Enforce extra security constraints to your API.
- Make accessible an extra capability that is provided in DataPower that is not yet accessible in the IBM API Connect policy catalog.

Note: For best results, do not try to update your existing user-defined policies, because those are already in use by your APIs. Instead, create a new version of the policy that you want to update, and then modify the API assembly to use the newer version of the policy.
See the following topics for more information about user-defined policies, and how to create and import them into IBM API Connect:

- **Authoring policies for the DataPower Gateway (v5 compatible)**
  A user-defined policy for the DataPower Gateway (v5 compatible) consists of a package that contains a YAML definition file that describes the policy, together with the policy implementation files. You import the package into one or more Catalogs in your IBM API Connect environment to make the policy available to APIs that are published to those Catalogs.
- **Authoring policies for the DataPower API Gateway**
  A user-defined policy for the DataPower API Gateway consists of a package containing configuration details that define the actions of the policy. You publish the package to the DataPower API Gateway to make it available to APIs that are deployed there.

Related information

- IBM API Connect Overview
- API policies

DataPower Gateway (v5 compatible) only

Authoring policies for the DataPower Gateway (v5 compatible)

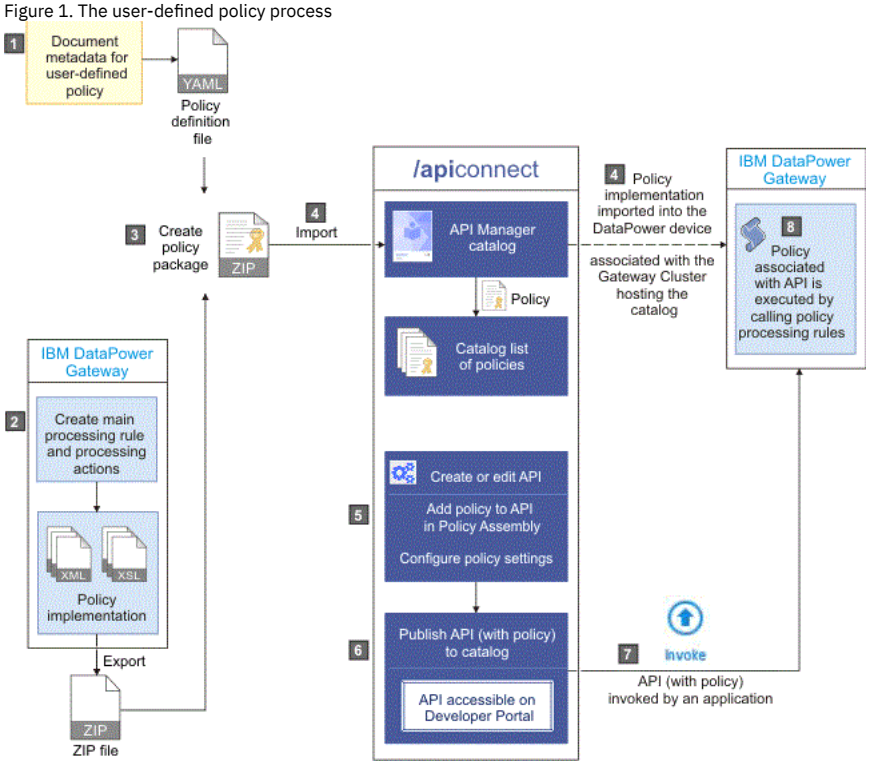
A user-defined policy for the DataPower® Gateway (v5 compatible) consists of a package that contains a YAML definition file that describes the policy, together with the policy implementation files. You import the package into one or more Catalogs in your IBM® API Connect environment to make the policy available to APIs that are published to those Catalogs.

How do user-defined policies work?

A user-defined policy is implemented as a DataPower processing rule. After a policy is imported into an IBM API Connect Catalog, the policy is available to be placed into an assembly flow of an enforced API. When the API is published, and invoked by an application, the API Gateway executes all policies that are associated with this API. An API Gateway that is running in DataPower calls the IBM DataPower Gateway processing rules that those policies implement.

The following diagram provides an overview of how to create and execute a user-defined policy in IBM API Connect.

Figure 1. The user-defined policy process



For details on how to define and deploy user-defined policies for the DataPower Gateway (v5 compatible), see the following subtopics:

- **Describing your policy**
  Describe your user-defined policy by creating a definition file in YAML format to document metadata about the policy.
- **Creating a new user-defined policy**
  Create a user-defined policy by configuring a policy definition file and its implementation, and then packaging the policy and importing it into IBM API Connect.
- **Packaging and importing your policies into IBM API Connect**
  Make your user-defined policy available to API developers by packaging it and importing it into an IBM API Connect Catalog.
- **Reference**
  Reference information for authoring policies in IBM API Connect.

DataPower Gateway (v5 compatible) only

Describing your policy

Describe your user-defined policy by creating a definition file in YAML format to document metadata about the policy.

About this task

A policy YAML file contains the following sections:

- A specification version.
- An information section.
- An attach section.
- A properties section.
- A gateways section.

You can create a policy YAML file by using any editor of your choice. Both .yaml and .yml file extensions are supported, but the use of the .yaml file extension is recommended by yaml.org.

Note: All keys and enumeration values that are specified in this topic are case-sensitive.

Procedure

The following steps describe how to construct a policy YAML file.

1. Set the specification version by adding the following line to the beginning of the file: `policy: 1.0.0`.
2. Complete the information section with details about the policy by using the following syntax:

```
info:
  title: Title of policy
  name: shortnameofpolicy
  version: 1.0.0
  description: An example policy
  contact:
    name: name1
    url: url1
    email: email1
```

where:

- **title** is the title of the policy. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- **name** is the short name for the policy and must contain only alphanumeric characters, the - (dash) character, and the _ (underscore) character (blank spaces are not supported). The name is case-sensitive, and should be 20 characters or less so that it can be displayed in the API Manager user interface. In the example, the name is shown as ***<policy-name>***.
  Important: The policy short name must be different from the OpenAPI names of the built-in policies, otherwise it will cause a policy conflict in the API assembly definition. For a list of the OpenAPI built-in policy names, see execute. Also, consider including an appropriate unique prefix or suffix string in the short names of your user-defined policies to prevent possible name conflicts with future built-in policies.
  Note: The **name:** property must be identical to the Node.js module name.
- **version** is the version number of the policy. This is a reserved property.
  Tip: The *version.release.modification* version numbering scheme is recommended, for example *1.0.0*.
  Note: The **info.version** in policy.yaml must match the **version** indicated in package.json.
- **description** (optional) is a short description of the policy.
- **contact** (optional) is the contact information for the policy, where:
  - **name** (string) is the identifying name of the contact person or organization.
  - **url** (string) is the URL that points to the contact information.
  - **email** (string) is the email address of the contact person or organization.

3. Complete the attach section by specifying which type of API flow the policy can be attached to.

```
attach:
 - rest
 - soap
```

where:

- **rest** indicates that this policy can be attached to a REST API flow.
- **soap** indicates that this policy can be attached to a SOAP API flow.

The attach section must contain at least one API flow type. If a policy can be attached to both API flow types, they must both be indicated in the attach section.

4. Complete the properties section by defining a JSON schema (based on JSON Schema Specification Draft 4, but rendered in YAML format) that contains the list of properties the policy will declare as required input. These properties are presented to the API author at development time, when the properties can be configured or mapped to values as required.
   The JSON schema defines a root object that contains the following JSON properties:
   - **type**
   - **properties**
   - **required**

   The syntax and definition of these properties matches the JSON schema definition. The following code block shows an example of a simple schema that defines one required property (*a_property*):

```
properties:
  $schema: "http://json-schema.org/draft-04/schema#"
  type: object
  properties:
    a_property:
    label: a label
    description: a description
    type: a type
  required:
    - a_property
```

   The root *type* of the schema must be an **object**, as shown in the example schema. You can define zero or more properties in the schema. Required properties are declared by using the **required** parameter, as shown in the example schema. The policy will not be accessible in the API Manager assembly editor, unless all the required properties have values. Each property is an object with the following items:
   - **label** is a short name for the property and is displayed in the Name column of the API Manager assembly editor.
   - **description** is a short description for the property and is displayed in the Description column of the API Manager assembly editor.
   - **type** must be one of the following primitives that are supported:
     - *integer*
     - *number*
     - *string*
     - *boolean* (this primitive is shown as a check box in the assembly editor)
     - *array*
   - **default** (optional) specifies a default value for the property.
   - **enum** (for properties with a **type** of *string*) is an array of valid values.
5. In your gateways section, specify that the policy is for the DataPower® Gateway.
   Enter the following code:

```
gateways:
  - datapower-gateway
```

6. Save the new policy as a YAML file using the value of the **name** property (in the **info** section) as the file's name.
   Required: To ensure a successful import, the YAML file must use the name specified in the policy's **name** property.

Results

You have created a user-defined policy definition YAML file that documents metadata about your policy.

Example

The following code block shows an example of a policy definition file that contains all the supported primitives. This policy must be saved as sampleimpl.yaml to ensure that it can be imported into API Connect.

```
policy: 1.0.0

info:
  title: Sample Policy
  name: sampleimpl
  version: 1.0.1
  description: This is a sample policy.
  contact:
    name: Steve Product Manager
    url: http://developer.acme.com/contacturl
    email: steve-product-manager@someemailservice.com

attach:
  - rest
  - soap

properties:
  $schema: "http://json-schema.org/draft-04/schema#"
  type: object
  properties:
    samplestring:
      label: String Property
      description: Sample string property
      type: string
    sampleboolean:
      label: Boolean Property
      description: Sample boolean property
      type: boolean
    sampleinteger:
      label: Integer Property
      description: Sample integer property
      type: integer
    samplenumber:
      label: Number Property
      description: Sample number property
      type: number
    samplestringwithenum:
      label: Enum Property
      description: Sample string enum property
      enum:
        - one
        - two
        - three
      default: one
      type: string
    samplearray:
      label: Array of properties
      description: Sample array of properties
      type: array
      items:
        type: object
        properties:
          string:
            label: String Property
            description: Sample string property in array
            type: string
          boolean:
            label: Boolean Property
            description: Sample boolean property in array
            type: boolean
        required:
          - string
  required:
    - samplestring
    - sampleboolean
    - sampleinteger
    - samplenumber
    - samplearray

gateways:
  - datapower-gateway
```

In the following code block, an example of a policy definition file for modifying message payload is shown.

```
policy: 1.0.0

info:
  title: Run GatewayScript
  name: gatewayscript-policy
  version: 1.0.0
  description: Execute GatewayScript
  contact:
    name: IBM DataPower Samples
    url: https://github.com/ibm-datapower/
    email: steve-product-manager@ibm.com

attach:
  - rest
  - soap
```

```
gateways:
  - datapower-gateway

properties:
  $schema: "http://json-schema.org/draft-04/schema#"
  type: object
  properties:
    source:
      label: "GatewayScript Source"
      description: "The location of the GatewayScript file to execute"
      type: string
      default: store:///identity.js
    value:
      label: "New Value"
      description: "The value to be added to the payload"
      type: string
      default: "Hello Policy"
required:
    - source
    - value
```

What to do next

Create an implementation for your user-defined policy by using DataPower processing rules and actions. For more information, see Implementing your policy.

Related concepts

- Authoring policies

DataPower Gateway (v5 compatible) only

Creating a new user-defined policy

Create a user-defined policy by configuring a policy definition file and its implementation, and then packaging the policy and importing it into IBM® API Connect.

Review the following topics to learn how to create a user-defined policy.
Tip: Some sample policies are available on GitHub, and these policies can be downloaded and adapted for use with IBM API Connect. For more information, see API Connect Policies on GitHub.

- **Implementing your policy**
  Create an implementation for your user-defined policy by using DataPower processing rules and actions.

Related information

- IBM API Connect Overview

DataPower Gateway (v5 compatible) only

Implementing your policy

Create an implementation for your user-defined policy by using DataPower® processing rules and actions.

Before you begin

You must create a policy YAML file before you create an implementation. For more information, see Describing your policy.

You can create an implementation for your policy by using the usual tools available to IBM® DataPower Gateway developers. However, the DataPower user interface is typically the tool that is used to create processing rules and actions.
Note: You must be skilled in DataPower tooling and concepts, before you can create a policy implementation.
In order for your processing actions to work well under the API Gateway configuration, and also to enable the use of contextual information that is relevant to the processing actions, a library of functions and templates is available that you can use to assist you in writing the XSLT or GatewayScript transformations. This library provides the following mechanisms:

- Access to input property values (input from the assembly editor).
- Access to context values (the API Gateway runtime context).
- Access to the runtime payload message and media-type.
- Ability to modify the payload message.
- Ability to end the policy execution with an error.

About this task

You create an implementation for your user-defined policy in the IBM DataPower Gateway. The implementation must adhere to the following conventions:

1. You must create a main processing rule, and this rule will be the starting point for the policy. The rule name must start with the name of the policy (the value of the `name` property in the "info" section of the policy YAML file), followed by -main; for example *checkmembership-main*.
2. There are no restrictions on what actions the processing rule can execute, on condition that the rule adheres to the naming convention in point 1. and the instructions that are detailed in this topic.
3. The names of the processing actions and all other objects also must start with the name of the user-defined policy (the value of the `name` property in the "info" section of the policy YAML file).
4. If your processing rule runs transformation actions that use XSLT or GatewayScript files, these files must be stored in the following location: local://policy/*<policy-name>*.
5. If you have certificate and key files that must be stored in the cert: folder, the names of these files also must start with the name of the user-defined policy (as defined in the policy YAML file).
6. If your policy is using GatewayScript transformations, your policy code must require `apim.custom.js`, for example:

```
var apic = require('local://isp/policy/apim.custom.js');
```

The following steps describe how to create an implementation for your user-defined policy:

- Create a main processing rule.
- Create one or more processing actions:
  - Configure access to input properties.
  - Configure access to the runtime context.
  - Configure access to the input payload.
  - Configure access to the HTTP headers.
  - Modify the payload in XSLT or gatewayscript.
  - Configure the implementation to produce error information.
  - Set variables.
- Export your policy implementation.

.

Procedure

1. Create a main processing rule that is called *<policy-name>*-main where *<policy-name>* is the name of your user-defined policy (the value of the `name` property in the "info" section of the policy YAML file).
   Note: You must specify the following processing rule settings:
   - Rule direction: Both Directions
   - Non-XML Processing: on
2. Create one or more processing actions.
   Each processing action should have a unique name and must start with the name of the user-defined policy (as defined in the policy YAML file).
   Note: Different processing actions in DataPower might require different releases of the DataPower firmware. When using the GatewayScript actions, the minimum DataPower firmware that should be used is version 7.2.0.
   The following processing actions are available:

   - Configure access to the input properties in XSLT (`policyProperties`), or in GatewayScript (`apic.getPolicyProperty`).
     If required by the properties that are defined in the policy YAML file, when the policy is attached to an API the API developer must enter the input settings or variables for a particular property. This function provides the mechanism to retrieve these input settings or variables at run time. For an example code snippet, see Access to input properties code snippet.

   - Configure access to the runtime context in XSLT (`getContext`), or in GatewayScript (`apic.getContext`).
     When an API is invoked, runtime information about the request can be accessed by using a user-defined policy, and this information is known as the runtime context. The function `getContext` provides the mechanism to access this runtime context. For an example code snippet, see Access to runtime context code snippet. For a complete list of context variables, see API Gateway context variables.

   - Configure access to the input payload in XSLT (`payloadRead`), or in GatewayScript (`apic.readInput(callback)`).
     Some policies require access to the input message or payload that is provided by the application. Accessing the input message or payload during an assembly flow can be difficult, as this information might change as the policies and the assembly steps run. This function provides a mechanism to get the correct input message or payload at the time of policy execution. For an example code snippet, see Access to input payload code snippet.

     This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.

     Note: The `payloadType()` function can be called to determine what type of payload (XML or JSONx) will be returned by the `payloadRead()` or the `apic.readInput(callback)` function.
   - Configure access to the HTTP headers in XSLT (`getContext`), or in GatewayScript (`apic.getContext`).
     The HTTP header information of a request can be retrieved from the request context. The HTTP header names are normalized to lowercase. The function `getContext()` provides the mechanism to access the HTTP header information. For an example code snippet, see Access to HTTP headers code snippet.

     Note: Access or modification of HTTP headers by using DataPower extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.
   - Modify the payload in XSLT or GatewayScript.
     When a policy implementation is required to change a payload message, you must configure the DataPower Transform Processing Action to set the Output context field to *OUTPUT* (referred to as the *OUTPUT* named context).

     The output must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template. For an example code snippet, see Modify the payload code snippet.

     Note: The modify the payload processing action works only with a proxy task. An HTTP or web service task always overwrites the output of a policy implementation.
   - Configure the implementation to produce error information.
     If you require your policy implementation to produce error information when there is a failure, you must configure the implementation by calling the error template. For an example code snippet, see Configure error information code snippet.

   - Set variables.

Use the **setVariable** template to set a runtime variable to a specified string value. This value can then be retrieved by using the function **getVariable()**, or by mapping the value into a step. For an example code snippet, see Set variables code snippet.

3. Export your policy implementation from DataPower.
   From the DataPower user interface, export the objects and files that are referenced by your main processing rule as a compressed file. No other objects or files should be exported. The following file list is an example configuration of an export from DataPower:

```
Archive: checkmembership.zip

11819    11-19-2014 22:53 dp-aux/basetypes.xml
3028207  11-19-2014 22:53 dp-aux/drMgmt.xml
6456     11-19-2014 22:53 dp-aux/map-dmz.xsl
7299     11-19-2014 22:53 dp-aux/management.xsl
23130    11-19-2014 22:53 dp-aux/SchemaUtil.xsl
216003   11-19-2014 22:53 dp-aux/clixform.xsl
5284     11-19-2014 22:53 local/policy/checkmembership/check.xsl
5061     11-19-2014 22:53 export.xml
```

   The example shows the exported configuration file (export.xml) that contains all the processing actions and DataPower objects, the referenced files, and the DataPower configuration schemas that are created by using the configuration export in the DataPower user interface.

Results

You have created an implementation for your user-defined policy and exported this implementation from the DataPower user interface.

What to do next

Create a user-defined policy package that can then be imported into IBM API Connect. For details, see Packaging and importing your policies into IBM API Connect.

Related concepts

- Authoring policies

Related information

- IBM DataPower Gateway

DataPower Gateway (v5 compatible) only

Packaging and importing your policies into IBM API Connect

Make your user-defined policy available to API developers by packaging it and importing it into an IBM® API Connect Catalog.

Before you begin

Before you can package a user-defined policy and import it into IBM API Connect, you must complete the following tasks:

1. Describe your policy in a YAML file.
2. Implement your policy by using DataPower processing rules and actions.

Procedure

To package and import your user-defined policy, complete the following steps:

1. Create a .zip file that contains the following folder structure:

```
<policy-name>.yaml
implementation/
              policy-name.zip
```

   where
   - *<policy-name>*.yaml is your policy definition file. The YAML file must use the name that is specified in its **name** property.
   - implementation/*policy-name*.zip is your policy implementation for a policy that is deployed to the DataPower Gateway. The policy implementation file contains the DataPower processing rules and actions that were exported from DataPower in step 2.

   The name of the .zip file must start with the name of the user-defined policy (as defined in the policy YAML file). If the implementation requires certificate and key files, these files must be added to the implementation directory.

   Note: Your package .zip file must contain a .zip policy implementation file.
2. Use the API Manager user interface to import your policy into a Catalog. For more information, see Importing a user-defined policy into a Catalog.
   Note:
   - You must import your user-defined policy into every Catalog in API Manager that you require your policy to run in.
   - For the policy to be displayed on the palette in the API assembly editor, you must import the user-defined policy into the Sandbox Catalog.

What to do next

Make your user-defined policy package accessible to the API Designer UI. To display user-defined policies in the API Designer UI, edit the .apiconnect/config file to point to the directory containing the policy.yaml file. You can edit the global config file (~/.apiconnect/config) or a project-specific config file (~/ProjectA/.apiconnect/config). Follow these steps:

1. Create one or more directories to hold your policy.yaml files that are accessible to Node.js, and copy the user-defined policy package into this directory. For example, create directories named /ProjectA/Policies and /ProjectB/Policies.
2. Extract the policy package so that the folder structure is visible, for example *policy*.yaml and implementation/*policy-name*.zip. The file containing the custom policies must be named policy.yaml.
3. Add the `userPolicies: [ ]` array object to the config file in ~/*ProjectA*/.apiconnect/ or ~/.apiconnect/, and edit the array object so it contains the absolute path to the director(ies) containing the policy.yaml file.
   Following is an example of an ~/.apiconnect/config file showing the userPolicies entry pointing to the directory containing the policy.yaml file:

```
apim_server: us.apiconnect.ibmcloud.com
us.apiconnect.ibmcloud.com.meta:
 formFactor: BLUEMIX_PUBLIC
 serverCapabilities:
    authTypes:
      - basic
      - token
 toolkit:
    version_recommended: 2.1.30
    version_minimum: 2.1.30
catalog: >-
  apic-catalog://us.apiconnect.ibmcloud.com/orgs/sampleorg-myspace2/catalogs/sb
app: >-
  apic-app://us.apiconnect.ibmcloud.com/orgs/sampleorg-myspace2/apps/acme-bank
userPolicies:
  - /ProjectA/Policies
```

   Note:
   - The `userPolicies: []` array object contains a list of absolute paths to the policy director(ies). The values can point to a parent location, for example `userPolicies: ["~/mypolicies"]`. Alternatively, each policy location can be specified in the array, for example `userPolicies: ["~/mypolicies/policya", ~/mypolicies/policyb"]`. However, do not use both of these approaches in the same directory. For example, do not create policies in a directory structure like this ~/mypolicies, ~/mypolicies/policya, and ~/mypolicies/policyb.
4. Open the API Designer UI. The user-defined policy is available on the policy assembly palette, and you can add the policy to your API definition.

DataPower Gateway (v5 compatible) only

Reference

Reference information for authoring policies in IBM® API Connect.

- **Implementation code examples**
  Example code snippets to help the creation of a user-defined policy implementation.

DataPower Gateway (v5 compatible) only

Implementation code examples

Example code snippets to help the creation of a user-defined policy implementation.

Note: If you are using GatewayScript, you must include the following command:

```
var apic = require(./apim.custom.js);
```

where *apic* is the common name used for the GatewayScript examples in this topic. However, *apic* could be any given name of your choice, for example you could use:

```
var apim = require(./apim.custom.js);
```

and then you would start your calls with `apim`.

- Access to input properties code snippet
- Access to runtime context code snippet
- Access to input payload code snippet
- Access to HTTP headers code snippet
- Modify the payload code snippet
- Configure error information code snippet
- Set variables code snippet

Access to input properties code snippet

The following code block shows an example of how to access the input properties by using the XSLT `policyProperties()` function. The example defines a property that is named `a_property`, which is declared as an integer value, but is retrieved in XSLT as text.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
```

```
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="p" select="apim:policyProperties()" />
    <xsl:message>
      The value of my input property is
      <xsl:value-of select="$p/a_property" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getPolicyProperty(propertyName)
```

where ***propertyName*** is the name of the input property that you want to access. If the input property name is blank, the action will return all input properties.

Access to runtime context code snippet

The following code block shows an example of how to access the runtime context by using the XSLT **getContext()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="client-id" select="apim:getContext('client.app.id')" />
    <xsl:message>
      The calling application is
      <xsl:value-of select="$client-id" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getContext(varName)
```

where ***varName*** is the name of the context variable that you want to access.
For a complete list of context variables, see API Gateway context variables.

Access to input payload code snippet

The following code block shows an example of how to access the input payload by using the XSLT **payloadRead()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="input" select="apim:payloadRead()" />
    <xsl:message>
      The input payload is
      <xsl:copy-of select="$input" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.readInput(callback)
```

A callback is required because the actual payload read is asynchronous. The callback method is called when the payload is ready.
This function returns an XML node-set that contains the payload of the request. If the payload is in JSON format, a JSONx node-set is returned that can then be manipulated within an XSLT or GatewayScript stylesheet. If the payload is not in JSON or XML format, the node-set that is returned is empty.
The following example shows how to use the **payloadType()** function to determine what type of payload (XML or JSONx) will be returned by the XSLT **payloadRead()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="payloadType" select="apim:payloadType()" />
    <xsl:message>
      <xsl:text>Payload type is [</xsl:text>
      <xsl:value-of select="$payloadType" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

Access to HTTP headers code snippet

The following code block shows an example of how to access the HTTP headers in XSLT by using the `getContext()` function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="content-type" select="apim:getContext('request.headers.content-type')" />
    <xsl:message>
      The request content type is
      <xsl:value-of select="$content-type" />
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

If you are using GatewayScript, you must call the function:

```
apic.getContext(request.headers.headerName)
```

where ***headerName*** maps to the name of the header you want to access.
Note: Access or modification of HTTP headers by using DataPower® extensions, such as `dp:set-request-header`, is not advisable, as such actions might yield unexpected results when the policy is combined with other policies and assembly steps.

Modify the payload code snippet

The output from a user-defined policy must be an XML node-set, which represents an XML or SOAP message, or a JSON message by using JSONx. The following code block shows an example of how to modify the payload in XSLT. To assist the API Gateway policy framework to accept the new or transformed message, call the `apim-output` template, as shown in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:apim="http://www.ibm.com/apimanagement"
  xmlns:jsonx="http://www.ibm.com/xmlns/prod/2009/jsonx">

  <!-- Contains the APIM functions -->
  <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <!-- Creates a JSON document (empty object is for simplicity) -->
    <jsonx:object>
    </jsonx:object>

    <!-- Indicates the media type of the output being produced -->

    <xsl:call-template name="apim:output">
      <xsl:with-param name="mediaType" select="'application/json'" />
    </xsl:call-template>
  </xsl:template>

</xsl:stylesheet>
```

where `mediaType`:

- `'application/json'` is when the output is written in JSONx format.
- `'application/xml'` is when the output is written in XML format.

If you are using GatewayScript, you must call the function:

```
apic.output(mediaType)
```

where ***mediaType*** is:

- **application/json** is when the output is written in JSONx format.
- **application/xml** is when the output is written in XML format.

Specifying the media type allows the next steps in the assembly flow to understand how to process the new payload.

Tip: The output from a user-defined policy must be XML or JSONx. JSONx is an IBM standard format to represent JSON as XML. One way to convert output GatewayScript JSON data into JSONx, is to add a **Convert Query Params to XML** action to follow the GatewayScript action within the same policy rule. The **Convert Query Params to XML** action must have an **Input Conversion** with the **Default Encoding** set to **JSON**. The output from the GatewayScript action must be the input for the **Convert Query Params to XML** action for JSONx to be produced.

Configure error information code snippet

The following XSLT code block shows an example of how to configure the policy implementation to produce error information by calling the **apim-error** template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:apim="http://www.ibm.com/apimanagement">

  <!-- Contains the APIM functions -->
  <xsl:include
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <!-- Indicates this policy has a failure and provides
       additional information for the client application -->
  <xsl:template match="/">
    <xsl:call-template name="apim:error">
      <xsl:with-param name="httpCode" select="'401'" />
      <xsl:with-param name="httpReasonPhrase" select="'Unauthorized'" />
      <xsl:with-param name="errorMessage" select="'Please select a Plan'" />
    </xsl:call-template>
  </xsl:template>

</xsl:stylesheet>
```

where:

- **httpCode** is the code of the required error message.
- **httpReasonPhrase** is the reason for the error.
- **errorMessage** is the suggested action for the user.

If you are using GatewayScript, you must call the template:

```
apic.error(name, httpCode, httpReasonPhrase, message)
```

where:

- **name** is the name of the error.
- **httpCode** is the code of the required error message.
- **httpReasonPhrase** is the reason for the error.
- **message** is the suggested action for the user.

Set variables code snippet

The following XSLT code block shows an example of how to set a runtime variable to a specified string value by calling the **setVariable** template.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:call-template name="apim:setVariable">
      <xsl:with-param name="varName" select="'serviceEndpoint'" />
      <xsl:with-param name="value" select="'https://endpoint.host.com/data'" />
    </xsl:call-template>
    <xsl:message>
      <xsl:text>Variable [</xsl:text>
      <xsl:value-of select="'serviceEndpoint'" />
      <xsl:text>] set to [</xsl:text>
      <xsl:value-of select="'https://endpoint.host.com/data'" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

where:

- **varName** is the name of the runtime variable that you want to set a value to.

- **value** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **value** as **$(request.headers.content-type)**.

If you are using GatewayScript, you must call the template:

```
apic.setvariable(varName, varValue, action)
```

where:

- **varName** is the name of the runtime variable that you want to set a value to, or that you want to add or clear.
- **varValue** is the string value that you want to set the variable to. This can be a literal value, or another variable. For example, to set your named variable to the value of the Content-Type header in a request, you would specify the **varValue** as **request.headers.content-type**. This property is only required when **set** or **add** is specified as the action.
- **action** is the action that you want to apply to the variable. Valid options are:
  - **set**
  - **add**
  - **clear**

  If no option is set, the default option of **set** is applied.

The following XSLT example shows how to retrieve the value of a runtime variable by using the **getVariable()** function.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  xmlns:func="http://exslt.org/functions"
  xmlns:apim="http://www.ibm.com/apimanagement" extension-element-prefixes="dp func apim">

  <!-- Contains the APIM functions -->
  <xsl:import
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.policy.doc_local:_isp_policy_apim.custom.xsl" />

  <xsl:template match="/">
    <xsl:variable name="varValue" select="apim:getVariable('serviceEndpoint')" />
    <xsl:message>
      <xsl:text>Variable [</xsl:text>
      <xsl:value-of select="'serviceEndpoint'" />
      <xsl:text>] = [</xsl:text>
      <xsl:value-of select="$varValue" />
      <xsl:text>]</xsl:text>
    </xsl:message>
  </xsl:template>

</xsl:stylesheet>
```

where

- **varValue** is the name of the runtime variable that you want to retrieve a value for.

If you are using GatewayScript, you must call the function:

```
apic.getvariable(varName)
```

where ***varName*** is the name of the runtime variable that you want to retrieve a value for.

Related concepts

- Authoring policies

Related tasks

- Implementing your policy

API Gateway only

Authoring policies for the DataPower API Gateway

A user-defined policy for the DataPower® API Gateway consists of a package containing configuration details that define the actions of the policy. You publish the package to the DataPower API Gateway to make it available to APIs that are deployed there.

There are two types of user-defined policies, *Catalog scoped* user-defined policies and *global scoped* user-defined policies:

Catalog-scoped user-defined policies
: Catalog-scoped user-defined policies are available to APIs only in the Catalogs that you specify. Use a Catalog scoped user-defined policy if you want to limit the availability of your policy on a Catalog specific basis. The possible actions of a Catalog scoped user-defined policy are limited to the API Connect built-in assembly policies. For more information, see Defining, packaging, and publishing a catalog-scoped policy for the API Gateway.

Global-scoped user-defined policies
: Global-scoped user-defined policies are available to APIs in every Catalog in every provider organization. Use a global scoped user-defined policy in the following situations:

  - You want to make your policy available everywhere rather than limiting its availability to specific Catalogs.
  - Your policy uses a DataPower implementation, where configuration changes are made directly on the DataPower API Gateway.

For more information, see Defining, packaging, and publishing a global-scoped policy for the API Gateway.

- **Defining, packaging, and publishing a catalog-scoped policy for the API Gateway**
  Create a catalog-scoped, user-defined policy that can be used in API assemblies within the catalog where the policy is published.
- **Defining, packaging, and publishing a global-scoped policy for the API Gateway**
  Create a global-scoped, user-defined policy that is available to APIs in any of the catalogs in any provider organization.

Defining, packaging, and publishing a catalog-scoped policy for the API Gateway

Create a catalog-scoped, user-defined policy that can be used in API assemblies within the catalog where the policy is published.

About this task

When a catalog-scoped policy is published, you can only use it in the API Assembler's Policy palette for APIs contained within the same catalog where the policy was published. Complete the following tasks to define, package, and publish the catalog-scoped policy.

Tip: During this task, you will create several zip files. On Mac OS X, run the `zip` command on the command line with the following flags:

```
-x ".DS_Store" -x "__MACOSX"
```

Including the flags prevents the auto-generated .DS_Store and __MACOSX files from being added to the zip.

- **Structure of the catalog-scoped user-defined policy YAML file**
  Review the required structure of the YAML file that contains your catalog-scoped user-defined policy.

Step 1: Define a catalog-scoped policy for API Gateway

Use the DataPower Web GUI to define a new policy that can be used in API assemblies within the catalog where the policy is published.

Procedure

1. Log in to the DataPower Web GUI.
2. Create a new "test" domain and then switch to it.
   This test domain will be used as a "sandbox" to build and export the custom policies.
3. Navigate to Assembly Function and create a new object:
   a. Name the object ***<org-name>_<catalogname>_udp-<policyname>_<version>***.
      Example: "testorg_stagecatalog_udp-testpolicy_1.0.0". Use the same name for all of the objects that you create in this task.
      Note: This naming convention specifies the provider organization and catalog where the policy will be available, as well as the policy's name and version.
   b. Under the "Assembly" property, create a new Assembly.
   c. Under the new Assembly, create a new Rule.
   d. Under the new Rule, create a new API Action.
   e. Optionally add an implementation file for the new Action.
      For example, an XSLT Action requires an XSL file. Create the file in the using the following path and name (where "udp" refers to user-defined policy:

      local:///*<udp-name>*/*<udp-name>*.*<file-extension>*

   f. Click Apply and then save all of the new objects.
4. Export the new user-defined policy.
   a. Navigate to the newly created Assembly Function.
   b. Click Export to export the new user-defined policy to a file named *<policyname>*_*<version>*.zip.
   c. Open the exported *<policyname>*_*<version>*.zip file and confirm that all referenced files are included.
   d. Delete the dp-aux folder from the zip.
      This optional folder requires a lot of storage space and might cause a problem when you package and publish the new policy in the next task.

Step 2: Package and publish a catalog-scoped policy for API Gateway

Decide how you want the new catalog-scoped policy to be managed, and then publish it to make it available to API assemblies within a particular catalog.

About this task

Be sure to complete Step 1: Define a catalog-scoped policy for API Gateway before attempting to package and publish the new policy.

There are two methods for packaging and publishing a user-defined policy, based on how you want the policy to be managed:

- Step 2, Option A: Package and publish a catalog-scoped policy directly in the DataPower configuration
- Step 2, Option B: Package and publish a catalog-scoped policy as a gateway extension that API Connect manages

Step 2, Option A: Package and publish a catalog-scoped policy directly in the DataPower configuration

Package and publish a catalog-scoped policy so it can be stored directly on the DataPower API Gateway.

About this task

When you store a catalog-scoped policy on the DataPower API Gateway, the policy exists in the gateway's startup domain configuration and will be unaffected by the API Manager lifecycle. If you want to store the catalog-scoped policy directly on DataPower, you do not need to package it for API Manager. Instead, complete the following steps.

Procedure

1. Import the custom policy created in Step 1: Define a catalog-scoped policy for API Gateway to the application domain for every DataPower node in the cluster.
2. Proceed to Step 3: Create and publish a policy.yaml file, where you will create a file that describes the new policy so that API Manager can support it.

Step 2, Option B: Package and publish a catalog-scoped policy as a gateway extension that API Connect manages

Package and publish a catalog-scoped policy as a gateway extension in API Connect.

Before you begin

The xml-manager must be enabled on the default domain of every gateway that will have a gateway-extension published to it.

About this task

When you publish a catalog-scoped policy as a gateway extension in API Connect, it is subject to the API Manager lifecycle. The individual assembly functions will not be added to the API Connect Gateway Service (only global-scoped policies are added to the service).

Procedure

1. Rename the *<policyname>_<version>*.zip file (that you created in Step 1: Define a catalog-scoped policy for API Gateway) to *<org-name>_<cat-name>_<policyname>_<version>*.zip.
   The new file name indicates the provider organization and the catalog where the policy will be used.

2. Create a gateway-extension manifest.json file as explained in Gateway extensions manifest.
   The manifest contains one or more entries. The following example contains a sample entry for the new catalog-scoped policy and references the new policy filename.

   ```
   {
       "filename": "<org-name>_<cat-name>_<policyname>_<version>.zip",
       "deploy": "immediate",
       "type": "dp-import",
   },
   ```

3. Create a new zip file called gateway-extension.zip that contains the following files:
   - manifest.json
   - *<org-name>_<cat-name>_<policyname>_<version>*.zip
4. Publish the new gateway extension using the API Conref Toolkit or the Cloud Manager UI:
   - API Connect Toolkit:
     Use the **apic gateway-extensions:create** command to publish the gateway extension as explained in the **apic gateway-extensions** topic.

   - Cloud Manager UI:
     Complete the following steps to publish the gateway extension from Cloud Manager:

     - Log in to Cloud Manager.
     - Click Topology > ⋮ > Configure gateway extensions > Upload gateway-extension.zip file and upload your new gateway-extension.zip file.
5. Verify that the new catalog-scoped policy is available in the API Assembler's Policy palette.
   a. Log in to API Manager and open the Catalog where you published the new catalog-scoped policy.
   b. Create an API.
   c. Open the API Assembler page and view the Policy palette; verify that your new catalog-scoped policy displays and can be added to the API.

Step 3: Create and publish a policy.yaml file

Create a file that describes the properties of the catalog-scoped policy, and then publish it to make it accessible to API Manager.

Procedure

1. Create a YAML file that describes the new catalog-scoped policy.
   a. Create a file named *<policyname>*.yaml.
   b. Paste the following content into the new file:

   ```
   attach:
   - rest
   - soap
   policy: 1.0.0
   info:
     title: <policyname>
     name: <policyname>
     version: 1.0.0
     description: <description-here>
   properties:
     $schema: 'http://json-schema.org/draft-04/schema#'
     type: object
     properties:
       exampleProperty:
         label: Test Property
         description: Enter any value
         type: string
       exampleProperty2:
         label: Another Test Property
         description: Enter any value
         type: string
     required:
       - exampleProperty2
   ```

```
gateways:
- datapower-api-gateway
```

c. Modify the content as needed to correctly describe your new catalog-scoped policy.
For more information, see Structure of the catalog-scoped user-defined policy YAML file.

d. Save and close the file.

2. Create a new zip file called *<policyname>*.zip that contains the new YAML file.
3. Publish the new policy using either the API Connect Toolkit or the API Manager:
   - API Connect Toolkit:
     Use the **apic policies:create** command to publish the gateway extension as explained in the **apic policies:create** topic.
   - API Manager:
     Complete the following steps to publish the new policy:
     - Log in to API Manager.
     - Click Manage Catalogs > Select Catalog > Gateways > *selected_gateway* > ⋮ > View policies and upload your new policy.zip file.
4. Verify that the new catalog-scoped policy is available in the API Assembler's Policy palette.
   a. In API Manager, select the Catalog where you published the new catalog-scoped policy.
   b. Create an API.
   c. Click ⚙ (next to Save) and for the Target Catalog, select the catalog where you published the new policy.
   d. Save the API.
   e. Open the API Assembler page and view the Policy palette; verify that your new catalog-scoped policy displays and can be added to the API.

Structure of the catalog-scoped user-defined policy YAML file

Review the required structure of the YAML file that contains your catalog-scoped user-defined policy.

The .yaml file for a catalog-scoped user-defined policy contains the following sections:

- Specification version
- Information
- Attach
- Gateways
- Properties
- Assembly section

Specification version

Set the specification version by adding the following line to the beginning of the file:

```
policy: 1.0.0
```

Information

Complete the information section with details about the policy by using the following syntax:

```
info:
  title: Title of policy
  name: shortnameofpolicy
  version: 1.0.0
  description: An example policy
  contact:
    name: name1
    url: url1
    email: email1
```

where:

- **title** is the title of the policy. Any string can be used, but the title should be kept short so that it can be displayed in the API Manager user interface.
- **name** is the short name for the policy. Must be a single word, and contain only alphanumeric characters, and the - (dash) and _ (underscore) characters. The name is case-sensitive, and should be 20 characters or less so that it can be displayed in the API Manager user interface.
  Important: The policy short name must be different from the OpenAPI names of the built-in policies, otherwise it will cause a policy conflict in the API assembly definition. For a list of the OpenAPI built-in policy names, see execute. Also, consider including an appropriate unique prefix or suffix string in the short names of your user-defined policy to prevent possible name conflicts with future built-in policies.
  Note: The **name:** property must be identical to the Node.js module name.
- **version** is the version number of the policy. This is a reserved property.
  Tip: The *version.release.modification* version numbering scheme is recommended, for example *1.0.0*.
  Note: The **info.version** in policy.yaml must match the **version** indicated in package.json.
- **description** (optional) is a short description of the policy.
- **contact** (optional) is the contact information for the policy, where:
  - **name** (string) is the identifying name of the contact person or organization.
  - **url** (string) is the URL that points to the contact information.
  - **email** (string) is the email address of the contact person or organization.

For example:

```
info:
  title: Custom Invoke Policy
```

```
  name: custom-invoke
  version: 1.0.0
  description: Invokes httpbin
  contact:
    name: IBM DataPower Samples
    url: 'https://github.com/ibm-datapower/'
    email: steve-product-manager@ibm.com
```

Attach

Complete the attach section by specifying which types of API flow the policy can be attached to.

- **rest** indicates that this policy can be attached to a REST API flow.
- **soap** indicates that this policy can be attached to a SOAP API flow.

The attach section must contain at least one API flow type. If a policy can be attached to both API flow types, they must both be indicated in the attach section.

For example:

```
attach:
  - rest
  - soap
```

Gateways

Specify the DataPower® API Gateway, as follows:

```
gateways:
  - datapower-api-gateway
```

Properties

Complete the properties section by defining a JSON schema (based on JSON Schema Specification Draft 4, but rendered in YAML format) that contains the list of properties the policy will declare as required input. These properties are presented to the API author at development time, when the properties can be configured or mapped to values as required. The JSON schema defines a root object that contains the following JSON properties:

- **type**
- **properties**
- **required**

The syntax and definition of these properties matches the JSON schema definition. The following code block shows an example of a simple schema that defines one required property (*a_property*):

```
properties:
  $schema: "http://json-schema.org/draft-04/schema#"
  type: object
  properties:
    a_property:
    label: a label
    description: a description
    type: a type
  required:
    - a_property
```

The root *type* of the schema must be an **object**, as shown in the example schema. You can define zero or more properties in the schema. Required properties are declared by using the **required** parameter, as shown in the example schema. The policy will not be accessible in the API Manager assembly editor, unless all the required properties have values. Each property is an object with the following items:

- **label** is a short name for the property and is displayed in the Name column of the API Manager assembly editor.
- **description** is a short description for the property and is displayed in the Description column of the API Manager assembly editor.
- **type** must be one of the following primitives that are supported:
  - *integer*
  - *number*
  - *string*
  - *boolean* (this primitive is shown as a check box in the assembly editor)
  - *array*
- **default** (optional) specifies a default value for the property.
- **enum** (for properties with a **type** of *string*) is an array of valid values.

For example:

```
properties:
  $schema: 'http://json-schema.org/draft-04/schema#'
  type: object
  properties:
    test:
      label: Test Property
      description: Enter any value
      type: string
  required:
    - test
```

Assembly section

The assembly section defines the assembly policy flow that will be activated when the user-defined policy is invoked. For example:

```
assembly:
  execute:
    - invoke:
        version: 2.0.0
        title: custom-invoke
        header-control:
          type: blacklist
          values: []
        parameter-control:
          type: whitelist
          values: []
        timeout: 60
        verb: keep
        cache-response: protocol
        cache-ttl: 900
        stop-on-error: []
        target-url: 'http://httpbin.org/headers'
```

You can determine the correct content of this section in either of the following ways:

- Define the assembly flow by using the graphical assembly editor in either the API Designer or API Designer user interface, then copy the **`assembly:`** section from the Source tab. For more information on defining an assembly flow in the user interface, see The Assemble view and API policies and logic constructs.
- Code the policies in the **`execute:`** subsection of the **`assembly:`** section by hand; for details, see execute.

The assembly section can contain only the API Connect built-in policies. However, you can include a gatewayscript policy that references native DataPower configuration.

Defining, packaging, and publishing a global-scoped policy for the API Gateway

Create a global-scoped, user-defined policy that is available to APIs in any of the catalogs in any provider organization.

About this task

When a global-scoped policy is published, you can use it in the API Assembler's Policy palette for APIs in any catalog belonging to any provider organization. Complete the following tasks to define, package, and publish the global-scoped policy.

Tip: During this task, you will create several zip files. On Mac OS X, run the **`zip`** command on the command line with the following flags:

```
-x ".DS_Store" -x "__MACOSX"
```

Including the flags prevents the auto-generated .DS_Store and __MACOSX files from being added to the zip.

Step 1: Define a global-scoped policy for API Gateway

Use the DataPower Web GUI to define a new policy that can be used in API assemblies in all catalogs belonging to all provider organizations.

Procedure

1. Log in to the DataPower Web GUI.
2. Create a new "test" domain and then switch to it.
   This test domain will be used as a "sandbox" to build and export the custom policies.
3. Navigate to Assembly Function and create a new object:
   a. Name the object ***`<policyname>_<version>`***.
      Example: "testpolicy_1.0.0". Use the same name for all of the objects that you create in this task. Because the policy will be available in all provider organizations and catalogs, there is no need to specify that information in the object name.
   b. Under the "Assembly" property, create a new Assembly.
   c. Under the new Assembly, create a new Rule.
   d. Under the new Rule, create a new API Action.
   e. Optionally add an implementation file for the new Action.
      For example, an XSLT Action requires an XSL file. Create the file in the using the following path and name (where "udp" refers to user-defined policy:

      local:///*<udp-name>*/*<udp-name>*.*<file-extension>*

   f. Click Apply and then save all of the new objects.
4. Export the new user-defined policy.
   a. Navigate to the newly created Assembly Function.
   b. Click Export to export the new user-defined policy to a file named *<policyname>*_*<version>*.zip.
   c. Open the exported *<policyname>*_*<version>*.zip file and confirm that all referenced files are included.
   d. Delete the dp-aux folder from the zip.
      This optional folder requires a lot of storage space and might cause a problem when you package and publish the new policy in the next task.

Step 2: Package and publish a global-scoped policy for API Gateway

Decide how you want the new global-scoped policy to be managed, and then publish it to make it available to API assemblies.

About this task

Be sure to complete the steps in Step 1: Define a global-scoped policy for API Gateway before attempting to package and publish the new global-scoped policy. There are two methods for packaging and publishing a user-defined policy, based on how you want the policy to be managed:

- Step 2, Option A: Package and publish a global-scoped policy directly in the DataPower configuration

- Step 2, Option B: Package and publish a global-scoped policy as a gateway extension that API Connect manages

Step 2, Option A: Package and publish a global-scoped policy directly in the DataPower configuration

Package and publish a global-scoped policy so it can be stored directly on the DataPower API Gateway.

About this task

When you store a global-scoped policy on the DataPower API Gateway, the policy exists in the gateway's startup domain configuration and will be unaffected by the API Manager lifecycle.

Procedure

1. Import the *<policyname>_<version>*.zip file containing the new global-scoped policy to the application domain for every DataPower node in the cluster.
   Complete the following steps on every DataPower node in the cluster--if you omit a node, then the policy will not be available in the API Assembly's Policy palette.
   a. On the DataPower node, navigate to the API Connect Gateway Service object.
   b. Under the "User-defined policies" property, add a new Assembly Function by importing the *<policyname>_<version>*.zip file containing your new global-scoped policy.
   c. Click Apply, and then Save.
2. Verify that the new global-scoped policy is available in the API Assembler's Policy palette.
   a. Log in to API Manager.
   b. Create an API.
   c. Open the API Assembler page and view the Policy palette; verify that your new global-scoped policy displays and can be added to the API.

Step 2, Option B: Package and publish a global-scoped policy as a gateway extension that API Connect manages

Package and publish a global-scoped policy as a gateway extension in API Connect.

Before you begin

The xml-manager must be enabled on the default domain of every gateway that will have a gateway-extension published to it.

About this task

When you publish a global-scoped policy as a gateway extension in API Connect, it is subject to the API Manager lifecycle.

Procedure

1. Make a copy of the *<policyname>_<version>*.zip file containing your new global-scoped policy and remove the ***<version>*** number from the file name.
   This name change ensures that the relative path of the file meets the requirement to match the following regular expression: **`implementation\/[\w_-]+\.zip`**.
   You can retain the copy of the file with its original name for tracking purposes.

2. Create a local folder named implementation and store the renamed zip file in that folder.
3. Create a YAML file that describes the new global-scoped policy.
   a. Create a file named *<policyname>*.yaml.
   b. Paste the following content into the new file:

   ```
   policy: 1.0.0
   info:
     title: <policyname>
     name: <policyname>
     version: 1.0.0
     description: <description-here>
   ```

   Only the **`policy`** and **`info`** sections are needed in the *<policyname>*.yaml for this task.

   c. Save and close the file.
4. Create a new zip file called *<policyname>_<version>*.zip that contains the following files:
   - The implementation folder
   - *<policyname>*.yaml
5. Create a gateway-extension manifest.json file as explained in Gateway extensions manifest.
   The manifest contains one or more entries. The following example contains a sample entry for the new global-scoped policy and references the original policy *<policyname>_<version>*.zip file that you created in Step 1: Define a global-scoped policy for API Gateway:

   ```
   {
       "filename": "<policyname>_<version>.zip",
       "deploy": "immediate",
       "type": "user-defined-policy",
   },
   ```

6. Create a new zip file called gateway-extension.zip that contains the following files:
   - manifest.json
   - *<policyname>_<version>*.zip
     This is the file that you created in step 4 of this task (it contains the implementation folder and the *<policyname>*.yaml file).

7. Publish the new gateway extension using one of the following methods:
   - API Connect Toolkit:
     Use the **`apic gateway-extensions:create`** command to publish the gateway extension as explained in the **`apic gateway-extensions`** topic.

   - Cloud Manager UI:
     Complete the following steps to publish the gateway extension from Cloud Manager:

- Log in to Cloud Manager.
- Click Topology > > Configure gateway extensions and upload your new gateway-extension.zip file.

8. Verify that the new global-scoped policy is available in the API Assembler's Policy palette.
   a. Log in to API Manager.
   b. Create an API.
   c. Open the API Assembler page and view the Policy palette; verify that your new global-scoped policy displays and can be added to the API.

Developer Portal: socialize your APIs

The Developer Portal serves as a comprehensive platform for facilitating API sharing with application developers.

In addition to allowing application developers to find and subscribe to APIs, the Developer Portal provides additional features including forums, blogs, comments, and ratings, together with an administrative interface for customizing the Developer Portal to your brand and theme. See the following topics for more information.

Using the Developer Portal
: As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.

Configuring the Developer Portal site
: Customize the look and feel of the Developer Portal site, and brand it for your organization.

Concepts in the Developer Portal
: Understand the various concepts and terminology that are referenced throughout the Developer Portal.

Getting started configuring the Developer Portal site
: A quick start guide to configuring your Developer Portal site.

Developer Portal tutorials
: Guided examples that take you through some of the most common user scenarios, as well as some of the more complex areas.

- **Using the Developer Portal**
  As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.
- **Configuring the Developer Portal site**
  Customize the look and feel of your Developer Portal site, and brand it for your organization.
- **Managing the Developer Portal by using the toolkit CLI**
  You can perform operations on your Developer Portal by using the API Connect toolkit CLI.
- **Troubleshooting guide for the Developer Portal**
  Use this guide to help you diagnose and resolve Developer Portal issues in IBM® API Connect. For example, why is the Developer Portal not connecting to the Management server, why is my site not created, and why can't I log in?
- **Migrating your Developer Portal from Version 5 to Version 10**
  Guidance on how to migrate a Developer Portal site from IBM API Connect Version 5, to IBM API Connect Version 10.
- **Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10**
  Information about the upgrade of the Developer Portal from Drupal 9 to Drupal 10.
- **Developer Portal tutorials**
  Tutorials for using the Developer Portal.

Using the Developer Portal

As well as enabling application developers to find and use your APIs, the Developer Portal also provides features such as API analytics, forums, blogs, and rating facilities.

The Developer Portal is a convenient place to share APIs with application developers. After a Developer Portal has been enabled through the API Manager, and one or more API Products have been published, application developers can browse and use the APIs. Typically, a Developer Portal is customized to fit the corporate branding and design requirements of a particular organization. For more information about configuring a Developer Portal site, see Configuring the Developer Portal site. However, when the Developer Portal is first enabled it can be used as is, for example for test and development purposes, or for internal use only.

The following topics provide information about the main API consumer concepts in the Developer Portal, and how an application developer can navigate the default site. However, bear in mind that the actual consumer experience might be very different depending on how the site is configured. For tutorials guiding you through enabling a Developer Portal site, and navigating and using APIs as an application developer, see Tutorial: Creating the Developer Portal and Tutorial: Creating Accounts and Applications on the Developer Portal.

- **Logging in as a consumer to the Developer Portal by using the CLI**
  Use the toolkit CLI to log in to the Developer Portal as a consumer.
- **Exploring APIs and Products in the Developer Portal**
  As an application developer, you can browse, test, and subscribe to APIs in the Developer Portal by exploring the available API Products.
- **Applications in the Developer Portal**
  You can view, edit, configure, and create applications in the Developer Portal.
- **Analytics in the Developer Portal**
  You can view analytics for APIs in the Developer Portal at the application and organization levels. This information is displayed in dashboard views that show the analytics metrics in the form of *visualizations* (represented as charts).
- **Consumer organizations in the Developer Portal**
  Consumer organizations provide a way of grouping application developers together. You can add, remove, and configure Consumer organizations in the Developer Portal.
- **User accounts, passwords, and support in the Developer Portal**
  You can change general elements such as user accounts and passwords in the Developer Portal.

Related information

- IBM API Connect overview

Logging in as a consumer to the Developer Portal by using the CLI

Use the toolkit CLI to log in to the Developer Portal as a consumer.

Before you begin

To log in as a consumer, you must have created a developer account in the Developer Portal as explained in the following tutorial: Tutorial: Creating Accounts and Applications on the Developer Portal.

About this task

You can use the toolkit CLI to log in to the Developer Portal as a consumer to view it and test how it behaves for your customers.

Procedure

1. Set up the toolkit as explained in Installing the toolkit.
2. Determine the name of your identity provider.
   When you log in from the CLI, you must provide the name of the realm (identity provider) that hosts your API Connect user account. If you don't know the name of your account's realm, you can look it up by using the following command.

   In the CLI, run the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --server consumer_endpoint_api --mode consumer --catalog catalog_name_or_id --org
   <provider_org_name_or_id> --fields registry_type,realm
   ```

   Where:
   - *consumer_endpoint_api* is the endpoint of the consumer API.
   - *catalog_name_or_id* is the name or ID of your catalog.
   - *<provider_org_name_or_id>* is the name of ID of your provider organization.

   For example:

   ```
   apic identity-providers:list -s consumer.abce-ci-15608-master.abce.ibm.com -m consumer -c sandbox -o org --fields
   registry_type,realm
   total_results: 1
   results:
     - registry_type: 'lur'
       realm: 'consumer:4da2f1b9-05c1-4506-9e2d-951198d19ad3:faa5577d-625c-4fc8-8d19-26077c8aff95/sandbox-idp'
   ```

   In the response, the realm consists of the following elements:

   ```
   consumer:<provider_org_name_or_id>:<catalog_name_or_id>/<catalog_identity_provider_name>
   ```

   For example, `consumer:myorg:mycatalog/sandbox-idp`.
   Note that *<catalog_identity_provider_name>* is typically made up of *catalog_name*-idp, for example sandbox-idp.

3. In the toolkit CLI, run the following command to log in to the Developer Portal:
   Command syntax:

   ```
   apic login --mode consumer --server consumer_endpoint_api --username username --password password --realm realm
   ```

   Where:
   - *consumer_endpoint_api* is the endpoint of the consumer API.
   - *username* is your API Connect username.
   - *password* is the password that corresponds to your API Connect username.
   - *realm* is the user registry realm for the specified *<username>*.

   Example:

   ```
   apic login --mode consumer --server consumer.9a6e-88fa49bc.eu-gb.apiconnect.cloud.ibm.com -u my_username -p Blue123 -r
   consumer:frankenstein:om-portal/portal-idp

   Logged into consumer.9a6e-88fb51bc.eu-gb.apiconnect.cloud.ibm.com successfully
   ```

Exploring APIs and Products in the Developer Portal

As an application developer, you can browse, test, and subscribe to APIs in the Developer Portal by exploring the available API Products.

Browsing available APIs
: In API Connect API providers package their APIs into Products to offer one or more APIs to application developers. The providers then use Plans to control access to APIs and to manage API usage. Products are packages that contain both the APIs and the accompanying Plans.
  From the Developer Portal home screen, click API Products to browse the available products. You can then select a Product to explore the available Plans and APIs further, including viewing API code snippets, and request and response examples.
  Note: If you have an account on the Developer Portal, you can specify a default programming language for the code snippets to display in by editing the Code Snippet Language in your account settings.

Subscribing to a Plan
: A Plan is a collection of API operations or subsets of operations from one or more APIs. In order to subscribe to a Plan, you must have an account on the Developer Portal and have created an App. For more information, see User accounts, passwords, and support in the Developer Portal and Applications in the Developer Portal.

Rate limits specify how many requests an App is allowed to make during a specified time interval. A Plan can have a shared rate limit for all the operations, or each operation can have a different rate limit. In addition, Plans can have multiple rate limits set per Plan and per operation, at second, minute, hour, day, and week time intervals. Plans can also have burst limits to prevent usage spikes that might damage infrastructure. Multiple burst limits can be set per Plan, at second and minute time intervals.

After you have identified the Plan that you want to use, click Subscribe and then select the App that you want to use with this Plan. If the Plan is not restricted, you can use it immediately. If the Plan is restricted, the subscription is shown as Pending Approval, and you cannot use the requested Plan until the administrator approves your request.

You cannot subscribe to a monetized Plan without first adding your payment information to your Consumer organization. You can register your payment information by editing the Billing tab for your Consumer organization. Note: you must either be the Consumer organization owner, or have `Settings: Manage` permissions for your organization to be able to edit the billing information. Consumer organizations can have at most one payment method. You can delete your payment method and add a different method, for example if your credit card expires. If you change your subscription to a Plan that has a different pricing structure, the subscription billing system will add a prorated amount to the next invoice of your subscription billing cycle.

Testing an API
: You can test an API in the Developer Portal by using the interactive API document test tool. If the API operation that you want to test doesn't require a client ID, then you can use the test tool without the need to log in. For more information, see Testing an API by using the Developer Portal test tool.

Calling an API in your application
: When you have subscribed to a Plan and you begin coding your application, you must retrieve the operation URL in order to call the API. You can retrieve the operation URL from the API overview page. For more information, see Calling an API.

- **How to use a URL to navigate directly to a Product or API in the Developer Portal**
  You can navigate to a specific Product or API in the Developer Portal by using ***`x-ibm-name:version`*** in the URL.
- **Testing an API by using the Developer Portal test tool**
  You can test the behavior of an API without the need to write any code by using the Developer Portal test tool. You provide the necessary API parameters within the test tool and click Invoke to see the response.
- **Calling an API**
  When you have a selected a Plan and you begin coding your application, you must retrieve the operation URL to call the API.

How to use a URL to navigate directly to a Product or API in the Developer Portal

You can navigate to a specific Product or API in the Developer Portal by using ***`x-ibm-name:version`*** in the URL.

To find a Product or API in the Developer Portal you typically navigate there by searching in the UI, clicking on the relevant Product, and then selecting the API. However, if you know the ***`x-ibm-name:version`*** of your Product or API, you can insert that information into the URL to go there directly.

To navigate directly to a specific API, use the following URL syntax:

`portal.site.url`*`/productselect/`*`api_x-ibm-name:version`

For example:

`https://my.portal.com/myorg/sandbox/productselect/petstore-header-clientid:1.0.0`

To navigate directly to a specific Product, use the following URL syntax:

`portal.site.url`*`/product/`*`product_x-ibm-name:version`

For example:

`https://my.portal.com/myorg/sandbox/product/my-product:1.0.0`

To navigate directly to a specific API within a specific Product, use the following URL syntax:

`portal.site.url`*`/product/`*`product_x-ibm-name:version`*`/api/`*`api_x-ibm-name:version`

For example:

`https://my.portal.com/myorg/sandbox/product/my-product:1.0.0/api/petstore-header-clientid:1.0.0`

Testing an API by using the Developer Portal test tool

You can test the behavior of an API without the need to write any code by using the Developer Portal test tool. You provide the necessary API parameters within the test tool and click Invoke to see the response.

Before you begin

To use the Developer Portal test tool with an API that requires an App client ID, you must first complete the following tasks:

- Create an App. For more information, see Registering an application.
- Subscribe to a Plan that contains the API that you want to test. For more information, see Exploring APIs and Products in the Developer Portal.

About this task

The Developer Portal test tool is an interactive API document test tool. If the operation that you want to test does not require a client ID, then you can use Developer Portal test tool without the need to sign in. However, if the operation that you want to interact with requires a client ID, then you must sign in to the Developer Portal first.

Using the Developer Portal test tool is subject to the rate limit that is applied to an operation or Plan. For example, if an operation has a rate limit of 10 requests per minute and you invoke the operation, the number of requests that can be made is reduced to nine. The limitations are triggered every time that you click invoke within the minute interval. This caveat affects the quota of the App that is selected to use with the Developer Portal test tool, but it does not affect the quota of the other Apps that are using the same operation or Plan.

To test an API in the Developer Portal, the Allow this API to be tested check box must be selected in the API Manager. For more information, see Creating API definitions.

Restriction:

- There are security mechanisms that prevent you using the Developer Portal test tool to test an Implicit or Authorization Code grant type in an OAuth provider API. Other grant types in the same OAuth provider API **can** be tested. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option. For more information, see Disabling test tool restrictions.
- You cannot use the Developer Portal test tool with suspended applications.
- You can only test an unenforced API if **`testable=true`**, and if the existing API implements CORS and is using HTTPS.

Note: GraphQL APIs can be tested from the Developer Portal user interface. However, not all authentication methods are supported. You must use one of the following authentication methods:

- Unsecured
- Client ID, passed in the header
- Client ID and secret, passed in the header

Also, **`cost`** metrics are not supported, and rate limits specific to GraphQL are not listed in Product plans.

Procedure

1. To use the Developer Portal test tool with an API that does not require the client ID of an App, complete the following steps:
   a. Click API Products.
      All of the APIs that can be used by application developers are displayed.
   b. Click the name of the API that you want to test.
   c. Select an operation and then select Try It.
   d. Click Try this operation.
   e. Supply the values for required headers or parameters.
   f. If the operation is secured with Basic Authentication, supply the credentials.
   g. Click Send Request.
      The result is displayed in the Response Body field. You can continue to test different parameter values as necessary.
      Note:
      The first time that you click Try It, you might be presented with a security error. Copy the URL from the Request URL field, open it in a browser window, and accept the security certificate. You are not presented with the security error again.

      Also, for performance reasons, if the payload is large, more than 1500 DOM highlighted elements, you get the response, but it is not pretty printed.
2. To use the Developer Portal test tool with an API that does require the client ID of an App, complete the following steps:
   a. Sign in to the Developer Portal.
   b. If you have not already done so, you must create an App so that you can test an API that requires a client ID. For more information, see Registering an application.
   c. Ensure that your App is subscribed to the Plan that contains the API that you want to test. For more information, see Exploring APIs and Products in the Developer Portal.
   d. Click API Products.
      All of the APIs that can be used by application developers are displayed.
   e. Click the *API* that you want to test.
   f. Select the **`Provide credentials for Client ID (API Key)`** operation and then select Try It.
      The API Key Identification window displays.
   g. Use the Register an application in order to select a client ID drop-down list to select an App to test the API with.
   h. Click Save credentials.
   i. If a client secret is required, enter the value in the Client Secret field.
   j. Find the operation that you want to test, then click Try It.
   k. Supply the required parameters and values.
   l. If the operation is secured with Basic Authentication, supply the credentials.
   m. Click Send.
      The result is displayed in the Response section. You can continue to test different field values as necessary.
      Note: The first time that you click Invoke, you might be presented with a security error. Click the link that is provided to accept the security certificate. You are not presented with the security error again.

Related tasks

- Calling an API by using CORS

Calling an API

When you have a selected a Plan and you begin coding your application, you must retrieve the operation URL to call the API.

About this task

Be aware of the following points when you call APIs in IBM® API Connect:

- In the HTTP response message for status code 200 the reason phrase is replaced with `OK`.

- API error messages are displayed in English only.

Procedure

To retrieve the operation URL, complete the following steps:

1. Click API Products, then select a product.
2. Click the API that you want to work with.
   The API overview page opens.
3. Select the operation you need and then copy the endpoint.
   This is the URL that the application calls, and its structure is defined:

Type of API	URL
REST APIs	`https://host/org/catalog/api/operation`
SOAP APIs	`https://host/org/catalog/api` for all operations in the WSDL.

   where:
   - *host* is the fully qualified host name of your gateway cluster.
   - *org* is the URL path of your organization.
   - *catalog* is the name of your Catalog.
   - *api* is the name of your API.
   - *operation* is the URL path of your operation.
4. Take note of any parameters, the request body, and the response body. Code your application to create the expected requests and handle the expected responses.

Depending on the Identify your application using setting for the API, you might have to provide a client ID, or client ID and client secret. To do this, complete the following steps:

5. To find the client ID, complete the following steps:
   a. Click Apps, then click the application name that you want to work with.
   b. Select the Show check box for Client ID.
      The client ID is displayed.
   c. Provide the client ID with the header parameter &client_id=
      For example, the URL used in the API might be:

      `https://host/org/catalog/api/quote?loanAmount=20000`

      but when you call it with a client ID of *1234*, change the URL to:

      `https://host/org/catalog/api/quote?loanAmount=20000&client_id=1234`

      Note: API Gateway only A client ID is generated automatically by API Connect when you register an application. However, if you specify a customized client ID, by using the CLI or REST API, then the length must not exceed 512 bytes, otherwise the gateway will reject the API request and return a `401` error.
6. The client secret is produced when you register an application. Provide the client secret with the query parameter &client_secret=. If you did not note the client secret when you registered the application, you must reset it; for information, see Managing applications.

The client ID, or client ID and secret can be logged along with the URL. Web servers usually log the URL in their access logs, which would give away the client secret. If you do not want to expose your client ID or secret in the URL, complete the following steps.

7. For the client ID, set the header, `X-IBM-Client-Id`, as part of the HTTP message that the application sends when it calls the API.
   An example URL statement might be:

   ```
   curl --header "X-IBM-Client-Id: 1234" https://host/org/catalog/api/quote?loanAmount=20000
   ```

   Note: API Gateway only A client ID is generated automatically by API Connect when you register an application. However, if you specify a customized client ID, by using the CLI or REST API, then the length must not exceed 512 bytes, otherwise the gateway will reject the API request and return a `401` error.
8. For the client secret, set the header, `X-IBM-Client-Secret`, as part of the HTTP message that the application sends when it calls the API.
   For example, the URL would be:

   `https://host/org/catalog/api/quote?loanAmount=20000`

   and set the following HTTP headers:

   ```
   X-IBM-Client-Id=1234
   X-IBM-Client-Secret=ABCD
   ```

What to do next

Monitor the API and application usage. For more information, see Managing applications.

- **Calling an API by using CORS**
  CORS (cross origin resource sharing) is a technique that allows calls to be made from code that is running in a browser to a third-party server (such as APIs running on an API Connect). These calls are, by default, not allowed as per the same origin security policy that is applied to the browser sandbox. Without CORS support, web developers are required to use more complex techniques such as server-side proxies.

Calling an API by using CORS

CORS (cross origin resource sharing) is a technique that allows calls to be made from code that is running in a browser to a third-party server (such as APIs running on an API Connect). These calls are, by default, not allowed as per the same origin security policy that is applied to the browser sandbox. Without CORS support, web developers are required to use more complex techniques such as server-side proxies.

About this task

Note: CORS support is available only on the DataPower® API Gateway.
API Connect Gateway servers support CORS to make it as easy as possible for web developers to use APIs within their web applications.
CORS is supported in the following browsers:

- Chrome 3+
- Firefox 3.5+
- Internet Explorer V11, or later
- Opera 12+
- Safari 4+

A CORS enabled browser automatically sends either a simple CORS request, consisting of the original request with the addition of the `Origin` header, or a preflight request followed by a simple CORS request.

An example CORS preflight request is as follows:

```
OPTIONS /org/env/api/resourceHTTP/1.1
User-Agent: useragent details
Access-Control-Request-Method: GET
Access-Control-Request-Headers: header names
Host: x.xx.xxx.xx
Origin: https://example.com
Accept: */*
```

You do not need to create CORS requests yourself, other than for testing or troubleshooting purposes.

A CORS response is received from the gateway; for example:

```
HTTP/1.1 200 OK
X-Backside-Transport: FAIL FAIL
Connection: Keep-Alive
Transfer-Encoding: chunked
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers:accept, accept-language, content-type, x-ibm-client-id
Access-Control-Allow-Methods: methods allowed on the resource
Vary: Origin
```

Applications in the Developer Portal

You can view, edit, configure, and create applications in the Developer Portal.

Before you can subscribe to a Plan that contains the API or APIs that you want to use, you must first register an application by creating an App. You can create more than one App, and these Apps can be edited and deleted as required.

To learn about applications in the Developer Portal, use the following topic links.

- **Registering an application**
  Before you can use an API, you must register your application to the Developer Portal.
- **Upgrading a Development application to Production status**
  If development application workflow is enabled for the Catalog associated with your Developer Portal, any new application that you register initially has Development status. When you have completed your application testing, you can request to upgrade your application to Production status.
- **Managing applications**
  You can show or reset a client ID for an application, verify or reset a client secret, and view the details of the APIs in the application. You can also unsubscribe from a Plan that the application is subscribed to.
- **Editing an application**
  You can edit applications in the Developer Portal. If a user has the correct permissions, they can edit the content of any application in their organization.
- **Deleting an application**
  You can delete applications in the Developer Portal.
- **Changing an application image**
  You can change the image for an application in the Developer Portal.
- **Verifying an application client secret**
  You can verify an application client secret in the Developer Portal

Registering an application

Before you can use an API, you must register your application to the Developer Portal.

About this task

When you register an application, you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client secret.

You can, optionally, add further client ID/client secret pairs to an application, any of which can be used to identify the application when calling an API.

Procedure

To register an application, complete the following steps:

1. Click Apps.
   The Apps page opens.
2. Click Create new app.
3. Optional: Specify a URL in the Application OAuth Redirect URL(s) field.
   You can specify multiple OAuth redirect URLs by clicking Add another item. If only one redirect URL is specified, and the application does not provide the `redirect_uri` in the OAuth request, then API Connect automatically uses the one redirect URL specified. However, if more than one redirect URL is specified, then the application must provide the `redirect_uri` in the OAuth request, or the OAuth request is rejected.
4. Click Submit.
   Your application is displayed.
5. The client secret is hidden; click the Show check boxes, make a note of your client secret because it is displayed only once.
   You must supply the client secret when you call an API that requires you to identify your application by using a client ID and client secret.
   Note: The client secret cannot be retrieved. If you forget it, you must reset it.
6. Optional: In the Certificate field, paste the public X509 certificate for your application, in PEM format.
   The Certificate field is available only if the APIs that have been published to the Developer Portal include at least one API that has been secured with TLS mutual authentication. You must complete this field if you want to call an API that has been secured with TLS mutual authentication. For more information, see Composing a REST API definition.

   These certificates are sent to the management server and to the gateway, and are used for certificate verification when the application calls an API that has been secured with TLS mutual authentication.

   Important: When you paste the certificate, ensure that it contains no line breaks, and that the `(-----BEGIN CERTIFICATE-----)` prefix and `(-----END CERTIFICATE----)` suffix are removed.
7. Optional: The client ID is hidden, to display the client ID for your application, click Subscriptions, then select the Show check box for Client ID.
   The client ID is displayed and can be hidden again by clearing the check box.
8. Optional: To verify your client secret, click Subscriptions, then click Verify, enter your client secret in the Secret field, then click Submit.
   You have confirmed whether your client secret is correct or incorrect.
9. To add an additional client ID and client secret to the application, complete the following steps:
   a. Click Subscriptions.
   b. Click Add alongside Credentials.
      The "Add new application credentials" window opens.
   c. Enter a name and an optional summary, and click Submit.
   d. Select the Show Client ID or Show Client Secret check box to display the client ID or client secret for the new credentials.
   e. To add a description to a set of client credentials, or to change the current description, click Update alongside the required credentials.
   f. To remove a set of client credentials from the application, click Delete alongside the required credentials.

   If you add additional credentials to an application, any of the associated client ID/client secret pairs can be used to identify the application when calling an API.
   Note: If you add two or more sets of client credentials to an application, OAuth tokens are not shared between them; each client credential set uses a different OAuth token.
10. To add an image, click Upload image from the application overflow menu.
    A new window opens; click Browse, select an image from your directory, and click Submit.
11. Optional: To change the URL that authenticated OAuth flows for this application should be redirected to, click Edit from the application overflow menu, then enter a URL in the Application OAuth Redirect URL(s) field.
12. Optional: To change the application name or description, click the Edit from the application overflow menu.

Results

Your application is registered.

What to do next

Select a Plan to use with your application, see Exploring APIs and Products in the Developer Portal.

Related information

- IBM API Connect overview
- Creating an API key security definition

Upgrading a Development application to Production status

If development application workflow is enabled for the Catalog associated with your Developer Portal, any new application that you register initially has Development status. When you have completed your application testing, you can request to upgrade your application to Production status.

Procedure

To request to upgrade a Development application to Production status, complete the following steps:

1. From the Developer Portal home page, click Apps.
2. Click the title of the application that you want to upgrade.
3. Click Upgrade to production, then click Request upgrade to confirm the upgrade request.

Results

An upgrade approval request is submitted. Your application continues to have Development status while the approval request is pending, and a Pending Upgrade label is displayed on the application details page. When the request is approved, you will receive a notification, and the application will move automatically to Production status.

If the request is rejected, the status of the application remains unchanged, and the Pending Upgrade label is removed.

Managing applications

You can show or reset a client ID for an application, verify or reset a client secret, and view the details of the APIs in the application. You can also unsubscribe from a Plan that the application is subscribed to.

About this task

On the application details page, you can display and reset a client ID and client secret. You might want to reset a client secret if it is compromised or you forget it. A client secret is only displayed once, so after you reveal it for the first time at a time of your choosing, you can use this window to verify that what you have is correct or reset it if necessary.

You can, optionally, add further client ID/client secret pairs to an application, any of which can be used to identify the application when calling an API.

Procedure

To manage an application, complete the following steps:

1. Click Apps.
   The Apps page opens.
2. Select the application that you want to work with from the displayed list.
3. Click Subscriptions.
4. The client ID is hidden; to display the client ID, select the Show check box. The client ID is displayed, and can be hidden again by clearing the Show check box.
5. To reset your client ID, click Reset.
   Warning: API calls made by the application with the existing client ID will fail.
6. If the client secret is compromised or you forget it, click Reset to generate a new value for the client secret.
7. If you are unsure whether the value that you have for the client secret is correct, you can click Verify, enter the value, and click Submit to confirm whether or not it is correct.
8. To add an additional client ID and client secret to the application, complete the following steps:
   a. Click Subscriptions.
   b. Click Add alongside Credentials.
      The "Add new application credentials" window opens.
   c. Enter a name and an optional summary, and click Submit.
   d. Select the Show Client ID or Show Client Secret check box to display the client ID or client secret for the new credentials.
   e. To add a description to a set of client credentials, or to change the current description, click Update alongside the required credentials.
   f. To remove a set of client credentials from the application, click Delete alongside the required credentials.

   If you add additional credentials to an application, any of the associated client ID/client secret pairs can be used to identify the application when calling an API.
   Note: If you add two or more sets of client credentials to an application, OAuth tokens are not shared between them; each client credential set uses a different OAuth token.
9. To view more details about an individual API, click the API name.
   The operations are listed.
10. To unsubscribe from a Plan, click Unsubscribe alongside the Plan name.

Related information

- IBM API Connect overview

Editing an application

You can edit applications in the Developer Portal. If a user has the correct permissions, they can edit the content of any application in their organization.

Procedure

1. From the Developer Portal home page, click Apps.
2. Select the target application.
3. Click Edit from the application overflow menu.
4. Make any required changes, then click Submit.

Deleting an application

You can delete applications in the Developer Portal.

Procedure

1. From the Developer Portal home page, click Apps.
2. Select the required application.
3. Click Delete from the application overflow menu.
4. Click Delete again to confirm.

Changing an application image

You can change the image for an application in the Developer Portal.

Procedure

1. From the Developer Portal home page, click Apps.
2. Select the required application.
3. Click Upload image from the application overflow menu.
4. Browse for the target image, then click Submit.
   You can remove the image by clicking Remove.

Verifying an application client secret

You can verify an application client secret in the Developer Portal

Procedure

1. From the Developer Portal home page, click Apps.
2. Select the target application.
3. Click Subscriptions.
4. Click Verify for the client secret.
5. Enter your secret string and click Submit.
   You receive a notification saying whether or not the secret that you entered is correct.

Analytics in the Developer Portal

You can view analytics for APIs in the Developer Portal at the application and organization levels. This information is displayed in dashboard views that show the analytics metrics in the form of *visualizations* (represented as charts).

Restriction:

- If the Catalog that is associated with this Developer Portal has two or more gateway services enabled then, for analytics data to be available, the gateway services must all be associated with the **same** analytics service. If the set of associated analytics services for the gateway services includes two or more different analytics services then the Developer Portal does **not** display analytics data. An analytics service is associated with a gateway service in the Cloud Manager user interface; see Associating an analytics service with a gateway service.
- If the visualizations in your dashboard views display no data, show data only for a time period in the past, or the Dashboard tab does not show up, a possible reason might be that access to analytics data within API Connect has been switched off. Contact your API provider for confirmation.
- What API consumer analytics are displayed in the Developer Portal depends on the visibility configuration settings. For more information, see Configuring analytics in the developer portal.

Use the following topic links to learn how to access the various analytics available to you in the Developer Portal:

- **Application analytics in the Developer Portal**
  From the Developer Portal, you can view interactive analytics information about all of the APIs used by an application.
- **Organization analytics in the Developer Portal**
  From the Developer Portal, you can view interactive analytic information about all of the APIs within an organization.

Application analytics in the Developer Portal

From the Developer Portal, you can view interactive analytics information about all of the APIs used by an application.

About this task

All members of the consumer organization can view analytics information relating to APIs used by an application.

Note: What API consumer analytics are displayed in the Developer Portal depends on the visibility configuration settings. For more information, see Configuring analytics in the developer portal.

Procedure

1. Click Apps from the Developer Portal dashboard.
2. Click the name of the application for which you want to view analytics information.
   From the Dashboard tab, you can see your application analytics including, API Stats, Total Calls, and Total Errors.

Organization analytics in the Developer Portal

From the Developer Portal, you can view interactive analytic information about all of the APIs within an organization.

About this task

All members of the consumer organization can view analytics information relating to the APIs that are used within an organization.

Note: What API consumer analytics are displayed in the Developer Portal depends on the visibility configuration settings. For more information, see Configuring analytics in the developer portal.

Procedure

1. Click your userName from the Developer Portal dashboard.
2. From the drop-down menu, select My organization.
3. Click the Analytics tab.
   From the Analytics tab, you can see your organization analytics including, API Stats, Total Calls, and Total Errors.

Consumer organizations in the Developer Portal

Consumer organizations provide a way of grouping application developers together. You can add, remove, and configure Consumer organizations in the Developer Portal.

Developer Portal users must be a member of at least one Consumer organization. A Consumer organization can have multiple members and multiple applications. Members must register an application before they can subscribe to an API. A Consumer organization owns all the applications that are created within it. The members do not own any applications. As long as there is one member of a Consumer organization, all applications continue unchanged.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.
To learn more about Consumer organizations in the Developer Portal, see the following topics.

- **Adding a Consumer organization from within the Developer Portal**
  You can create a new Consumer organization from within your user account in the Developer Portal. The new Consumer organization appears also in the API Manager UI after it is created.
- **Editing the name of a Consumer organization**
  You can edit the name of your Consumer organization from within the Developer Portal.
- **Switching Consumer organizations**
  You can switch from one Consumer organization to another in the Developer Portal.
- **Adding users to a Consumer organization**
  If the API Provider organization administrator has granted you permission to collaborate, you can invite users to join your Consumer organization. Those organization members can then access the Developer Portal and use the APIs that have been made available to the Consumer organization.
- **Removing a user from a Consumer organization**
  You can remove a developer from a Consumer organization in the Developer Portal.
- **Changing the ownership of a Consumer organization**
  You can change the ownership of a Consumer organization in the Developer Portal.
- **Deleting a Consumer organization**
  You can delete a Consumer organization in the Developer Portal.

Adding a Consumer organization from within the Developer Portal

You can create a new Consumer organization from within your user account in the Developer Portal. The new Consumer organization appears also in the API Manager UI after it is created.

Procedure

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page.
3. Select Create organization from the drop-down menu.
4. In the Organization Title field, type the name of the new Consumer organization, then click Submit.

Results

Your new Consumer organization is created, and you can switch to it by clicking on its name in the Organization drop-down menu on the home page.

Editing the name of a Consumer organization

You can edit the name of your Consumer organization from within the Developer Portal.

Procedure

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page, and click My organization.
3. In the Manage tab, click the vertical ellipsis Manage organization icon, and click Edit organization.
4. Enter the new name for your Consumer organization in the Organization Title field, and then click Submit.
   You changed the name of your Consumer organization.

Switching Consumer organizations

You can switch from one Consumer organization to another in the Developer Portal.

Procedure

1. If you are a member of more than one Consumer organization, click the menu of Consumer organizations in the Developer Portal home page.
2. Select the Consumer organization that you want to switch to, and ensure that any applications you register are registered to the correct catalog.

Adding users to a Consumer organization

If the API Provider organization administrator has granted you permission to collaborate, you can invite users to join your Consumer organization. Those organization members can then access the Developer Portal and use the APIs that have been made available to the Consumer organization.

About this task

Depending on the permissions enabled by the Provider organization administrator, members can view applications and application activity, create and edit applications, manage client keys and subscribe to API Plans, or perform all activities.
Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.

Procedure

To invite users to join your Consumer organization, complete the following steps.

Note: The ability to invite users is only visible if the API Provider organization administrator has granted you permission to collaborate.

1. Sign in to your user account in the Developer Portal.
2. Click the arrow next to your Consumer organization name on the Developer Portal home page, and click My organization.
   The organization page is displayed.
3. On the Manage tab, select Invite.
4. Enter the email address of the new user in the Email field.
5. Under the Assign Roles heading, use the radio buttons to select a role for the new user, the roles are:
   - **Administrator**-- Can administer the Consumer organization, including invite new members, and can create, and edit applications, manage client keys, and subscribe to API Plans.
   - **Developer**-- Can create and edit applications, manage client keys, and subscribe to API Plans.
   - **Viewer**-- Can only view applications and application activity.

   For full details of the permissions that are assigned to the Developer Portal roles, see API Connect user roles.
6. Click Submit.
   An invitation is sent to the new user.

Results

The user is added to the Consumer organization list of members, and they are sent an email inviting them to join the organization. The user must click the link that is provided in the email to activate their account, and complete the setup.

Removing a user from a Consumer organization

You can remove a developer from a Consumer organization in the Developer Portal.

Before you begin

You must be the owner of your Consumer organization and have previously added the developer you wish to remove from the organization.
Note that removing a user from a Consumer organization does not delete their user account, it only removes them from that organization.

Procedure

1. Click your user name in the Developer Portal home page.
2. Click My organization in the menu.
3. Click Remove user for the user you want to remove.

Changing the ownership of a Consumer organization

You can change the ownership of a Consumer organization in the Developer Portal.

Before you begin

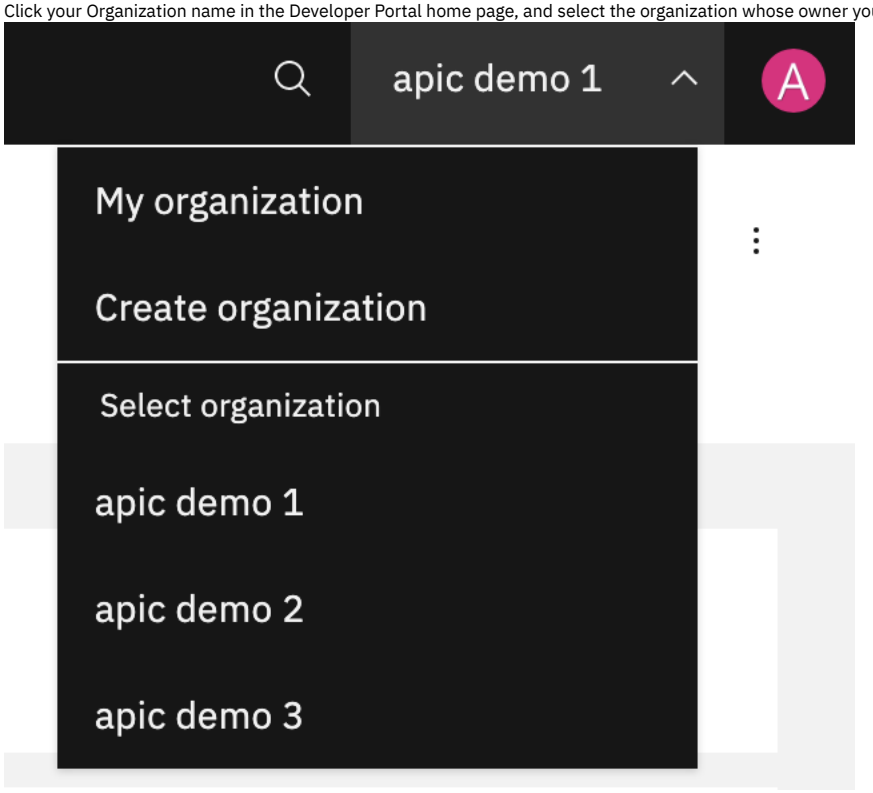
You can change the ownership of a Consumer organization only if you are the owner of that organization. The user that you transfer ownership to must already be a member of the organization. To have an account on a Developer Portal, you must be a member of at least one Consumer organization.

About this task

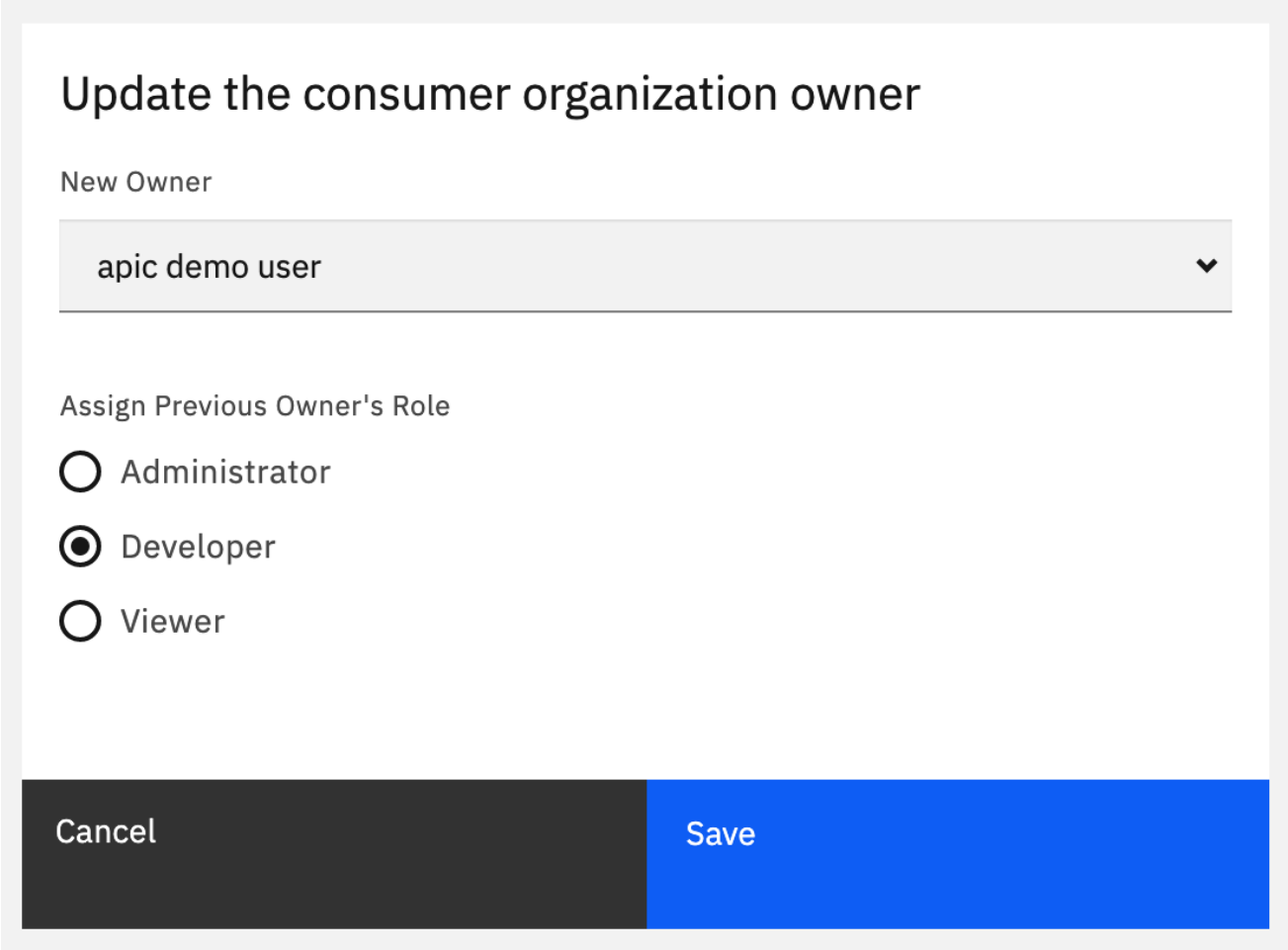
Transferring the ownership of a Consumer organization to another organization member, changes your role in that organization to App Developer.

Procedure

1. Click your Organization name in the Developer Portal home page, and select the organization whose owner you want to change by using the drop-down arrow.



2. Click your Organization name in the Developer Portal home page, and select My organization.
3. Click the options icon, ⋮ and select Change ownership.



4. Select the **`New Owner`** by using the drop-down arrow.
5. Use the radio buttons to **`Assign Previous Owner's Role`**, then click Submit.

Results

You successfully transferred ownership of a Consumer organization in the Developer Portal.

Deleting a Consumer organization

You can delete a Consumer organization in the Developer Portal.

Before you begin

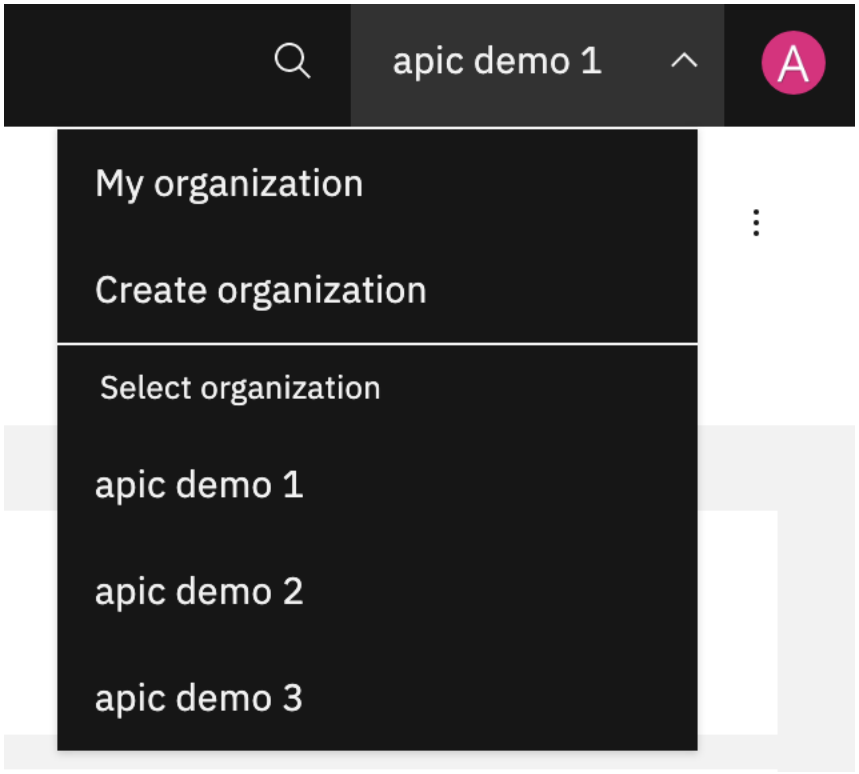
You can delete a Consumer organization only if you are the owner of that organization. To have an account on a Developer Portal, you must be a member of at least one Consumer organization. Therefore, if you are a member of only one Consumer organization, to delete that organization you must first either join or create another Consumer organization, or you can delete your whole developer account. For more information, see Deleting your developer account.

About this task

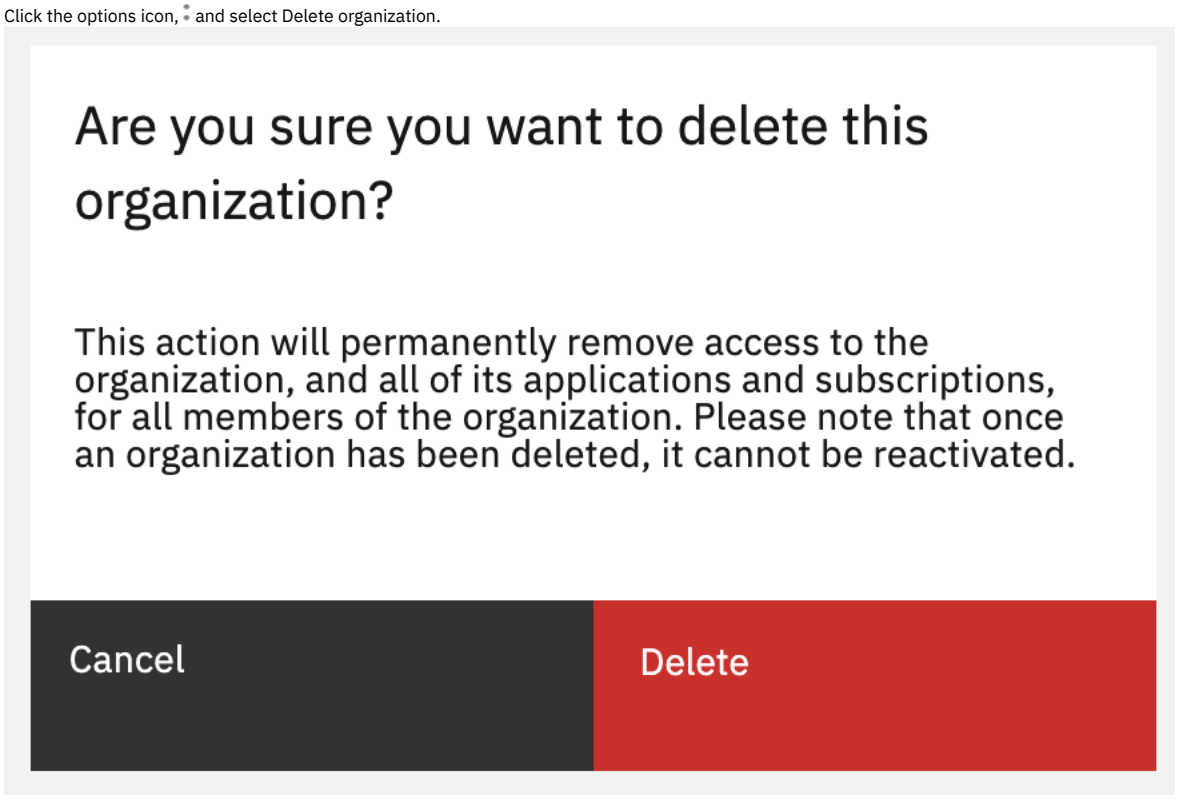
Deleting a Consumer organization in the Developer Portal permanently removes access to the organization, and all of its applications and subscriptions, for all members of the organization. Consider carefully the impact on any members of your Consumer organizations before you delete the organization. Any users that were members of only the deleted organization, must create a new Consumer organization when they next logon to the Developer Portal.
Important: When an organization is deleted, it cannot be reactivated. You might want to consider changing the ownership of a Consumer organization, rather than deleting it. For more information, see Changing the ownership of a Consumer organization.

Procedure

1. Click your Organization name in the Developer Portal home page, and select the organization that you want to delete by using the drop-down arrow.
   You must be in more than one Consumer organization.



2. Click your Organization name in the Developer Portal home page, and select My organization.
3. Click the options icon, ⋮ and select Delete organization.



4. Click Delete.

Results

You permanently deleted a Consumer organization in the Developer Portal.

User accounts, passwords, and support in the Developer Portal

You can change general elements such as user accounts and passwords in the Developer Portal.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.
To learn about general user features of the Developer Portal, see the following topics:

- **Creating a new developer account**
  You can create new developer accounts, and their associated Consumer organizations, in the Developer Portal.
- **Changing your account settings**
  You can edit your user account settings in the Developer Portal, such as changing your name, password, time zone, and language settings.
- **Deleting your developer account**
  You can delete your account in the Developer Portal.

Creating a new developer account

You can create new developer accounts, and their associated Consumer organizations, in the Developer Portal.

Before you begin

You need to provide an email address to create a new account. That email address must be a valid email address, and the domain that is used is required to have a valid MX record that is determined by DNS. You can disable the security module that checks that the email domain has a DNS record.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Extend > Disable module.
4. In the search bar enter `Check DNS`, then select the Check DNS module and click Disable.
5. Click Disable to confirm that you want to disable the module.

About this task

In order to use the Developer Portal, you must have a developer account, and be a member of at least one Consumer organization. When you create a developer account, a Consumer organization is created at the same time. You can be an owner, and a member of, multiple Consumer organizations. For more information about Consumer organizations, see Consumer organizations in the Developer Portal.
Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.

Procedure

1. From the Developer Portal home page, click Create account.
2. Populate the available fields with the details of the developer account and Consumer organization that you want to create.
3. Click Sign up.
   The email address that you provided for the new developer account is sent an email that contains an activation link for the new developer account. Use this link to activate your new account.

Changing your account settings

You can edit your user account settings in the Developer Portal, such as changing your name, password, time zone, and language settings.

About this task

You can edit various details about your user account, for example, change your password, update your code snippet language, change your time zone, select your preferred site language, and select an avatar.

Procedure

1. Click your user name in the Developer Portal home page, and click My account.
2. To update your password, click Change Password, and update the fields as required.
3. To update other elements of your account, such as your name, time zone, and language, click Edit and update the fields as required.
4. Click Save to save your changes.

Deleting your developer account

You can delete your account in the Developer Portal.

Before you begin

You must be the owner of the developer account that you want to delete, and you must not own more than one Consumer organization. If you do own more than one Consumer organization, you must either change the ownership of those organizations, or delete them, before you can delete your account. If you own only one Consumer organization, when you delete your account the Consumer organization is deleted at the same time as your account.

About this task

When you delete your account, any Consumer organizations that you own must also be deleted or transferred to new owners. If there are other members in a Consumer organization that you delete, these members will no longer have access to that organization, or any of its applications and subscriptions. Consider carefully the impact on any members of your Consumer organizations before you delete your account. For more information, see Deleting a Consumer organization.
Important: When an organization has been deleted, it cannot be reactivated. You might want to consider changing the ownership of your Consumer organizations before you delete your account. For more information, see Changing the ownership of a Consumer organization.

Procedure

1. Click your user name in the Developer Portal home page, and select My account.
2. Click the Edit tab.
3. Click Delete account.
4. Click Delete to confirm deletion of the account.
   This action deletes your developer account and, if you own a Consumer organization, this organization is also deleted.

Results

You successfully deleted your developer account in the Developer Portal.

Configuring the Developer Portal site

Customize the look and feel of your Developer Portal site, and brand it for your organization.

In IBM® API Connect Version 10 the Developer Portal is based on the open source Drupal 10 content management system, and consequently it is almost infinitely customizable. What an individual user is able to configure, depends on the role the user has been assigned, and the associated permissions of that role. A user can be assigned one of four Developer Portal roles:

Authenticated user
: An authenticated user is, typically, an application developer, and can perform all of the actions that are described in the Using the Developer Portal section.

Forum Moderator, Content Author, or Administrator
: In addition to performing all of the tasks that an Authenticated user role can, a Forum Moderator, Content Author, or Administrator role have additional permissions related to configuring the Developer Portal site. For more information, see Working with roles in the Developer Portal.
  Note: By default, the Developer Portal administration operations are available only if you have Administrator access.

Before you start to configure your Developer Portal site, you should familiarize yourself with the various concepts and terminology that is referenced throughout the Developer Portal. See Concepts in the Developer Portal.

When you are ready to start configuring your Developer Portal site, see Getting started configuring the Developer Portal site for a quick start guide on customization. There is also a tutorial that you can follow that takes you through an example customization scenario; see Tutorial: Creating a custom theme for the Developer Portal.

If you are migrating from Version 5 to Version 10, see Migrating your Developer Portal from Version 5 to Version 10 for guidance about how to transfer any specific customizations, such as custom modules and themes, from the Drupal 7 baseline to the Drupal 10 baseline.

In addition to the previous options, Developer Portal configuration operations are described in detail in the following sections:

Appearance
: Control and configure the general appearance of your Developer Portal site by setting and configuring a default theme.
  Important: Editing API Connect themes, modules, or Drupal core on the filesystem is not permitted or supported. For more information, see Creating a sub-theme and Extend.

Content
: Control multiple content entities in the Developer Portal, including the customization and restriction of specific content.

Structure
: Control the layout of the Developer Portal, such as configuring the front page, adding and changing blocks, and displaying Products and APIs in categories.

Configuration
: As an administrator, you can control multiple aspects of the Developer Portal configuration, including managing security and general configuration tasks.

People
: You can manage and customize the user experience in the Developer Portal, by altering permissions and assigning roles to specific users.

Forums
: Create and control forums in the Developer Portal.

Reports
: You can view reports about the general configuration and status of your Developer Portal site.

Extend
: You can extend the Developer Portal by creating and installing custom modules, or installing additional Drupal contributed modules.

Related information

- Drupal website

Concepts in the Developer Portal

The Developer Portal is built on Drupal 10, an open source content management technology. A good understanding of Drupal concepts and terminology enhances your ability to work with the Developer Portal.

In Drupal, most content on the website is treated as variations on the concept of a **node**. Node types, such as static pages, blog posts, and news items, are all stored in the same way. The navigation structure of the site is designed separately by editing **menus**, **views**, and **blocks**. Finally, the **theme** of the system controls the overall appearance of the site to visitors. Access to these depends on the permissions that your role possesses. These are explained more fully in the following sections.

Nodes

A node is a set of individual related content such as a page, a poll, an article, a forum topic, or a blog post. For example, when you create a blog post, and define the body text, you also define its title, content, author link, creation date, and taxonomy . Some of these elements are shown by the theme layer when the node is displayed, and some elements are metadata that control where and when the node is displayed at all. Each set of content in a Developer Portal site is a node. You can apply new features or changes to all content of a single node type.

Fields

A field is somewhere that you can add extra metadata to a node. The following types of data are examples of fields:

- Title
- Body
- Comment body
- Tags
- Image

You can create different types of fields, and they can be defined at the content-type level for content items and comments, at the vocabulary level for taxonomy terms, and at the site level for user accounts. Field types are defined by modules, and managed by the Field module. In addition, other modules might enable fields to be defined for their data.

Content types

A content type is a predefined collection of fields. Content types define the default fields that content editors use to add content in a Developer Portal site, and help structure the authoring and developing of content. Content types can be displayed in the Developer Portal. You can control which content types, and the order and format that they are displayed, in the Developer Portal.

Themes

You can use themes to control the overall appearance of your Developer Portal site. You can modify the appearance of the theme in following ways:

1. Extend the code of an existing theme.
2. Identify and use a theme that is provided by the Drupal community or a third-party website, and modify the theme settings.
3. Create a complete custom theme from scratch.

However, you cannot directly edit the API Connect theme as editing is not supported. Any edited versions of these files are overwritten when a fix pack or iFix is installed. So, the way to create a custom theme is to create a custom subtheme of the standard API Connect theme that the Developer Portal uses by default. A subtheme inherits all of the settings of its parent theme, apart from the settings that are overridden. For more information, see Creating a sub-theme.

Most themes use the Twig template engine for PHP.

Regions

The specific areas of a Developer Portal site in which content can be placed. Regions are customized and styled in the theme.

Blocks

Blocks are boxes of content that can be displayed in regions on your Developer Portal page. Blocks can be made available to your Developer Portal site by enabling specific modules. After a block is created, its appearance can be modified. You can also define which Developer Portal page or pages blocks appear on. Some modules can provide multiple blocks when they are enabled, while other modules might not define new blocks. For more information, see Adding and changing the blocks displayed on Developer Portal pages.

Modules

Modules are similar to the concept of plug-ins, in that they extend the core functions of the Developer Portal site. A set of modules is implemented by default with the Drupal core, and there are extra modules that can be enabled to extend the default functions. You can find and add more modules to your Developer Portal site from the internet. For more information, see Extend.

Views

You can use views to manipulate the content that is displayed on a page, block, and other visual elements in the Developer Portal site. Views can be used along with content types to format the appearance of your site to your specification. For more information, see Using the Views module in the Developer Portal.

Pages

Pages are used by the features that can modify the appearance of the Developer Portal to customize the appearance of your Developer Portal site. Pages can be customized to be as specific as you require, and can be configured to satisfy the context of the situation that they are used in.

Users

After you log in to the Developer Portal site, you have a user record in the Developer Portal database. User ID 1 is always reserved for the administrator account. Regardless of whether other user accounts are using remote authentication such as LDAP, the administrator account is always a local account to enable administration of the Developer Portal site. For more information, see People.

Roles

A role is a collection of permissions that define the actions that a user can do in the Developer Portal site. Users can be given one or more roles. By default, a user that is logged in to the Developer Portal is in the Authenticated role. Other roles that can be assigned to a user include:

- Administrator
- Forum Moderator
- Content Author

You can also create new custom roles. For more information, see Working with roles in the Developer Portal.

Permissions

Permissions define the actions that a user can or cannot do in a Developer Portal site. Permissions are additive. If a permission that enables a user to do an action is not assigned, the user cannot do the action. If a user has multiple roles, and any of them contain a specific permission, the user is able to do that action. There are also permissions that have security implications, and you are recommended to assign those permissions to trusted roles. For more information, see Controlling access to Developer Portal content.

Templates

Template files define how the output of a particular component can look. They are formatted as Twig template files. The following are examples of where you can use templates:

- html.html.twig - the template for Developer Portal HTML pages
- page.html.twig - the body of Developer Portal HTML page
- node.html.twig - the template for all of the content nodes
- comment.html.twig - the template for all of the comment nodes
- search-result.html.twig - the template for search results
- node--product.html.twig - the template for nodes that use the Product content type
- node--product--teaser.html.twig - the template for the previews for the Product content type

For more information about overriding templates in the Developer Portal, see Applying a modified content type template.

Related information

- Understanding Drupal
- Theming Drupal

Getting started configuring the Developer Portal site

A quick start guide to the main tasks involved in configuring the Developer Portal to your brand or theme.

The Developer Portal is based on the open source Drupal 10 content management software, and consequently is almost completely customizable. The following sections take you through the most common tasks to get started with branding the Developer Portal to your organizational requirements, including links to the related topics in the main table of contents. Many more customizing options are described in the main table of contents, and the best way of finding help with a specific customization in this content, is to use the documentation search bar.
Important:
All of the customizations that are detailed in this topic are preserved when you update to new versions of API Connect. Drupal updates, as well as new functionality, defect fixes, and security updates, are made available through API Connect update packages as determined by IBM. These update packages can be applied by upgrading your deployment; see the upgrade instructions for your platform in Installing and maintaining your IBM API Connect cloud. To find out what update packages are available, select your product on IBM Fix Central.

Directly editing any API Connect themes, modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when an update package is installed. Customizations must be done by using the proper mechanisms as described in this topic, such as custom modules and themes, and not by directly editing the source.

For more information about the API Connect support policy, see IBM® API Connect Support Lifecycle Policy.

- Create a custom theme
- Changing the page layout
- Changing the front page
- Advanced theme development
- Custom modules
- Hiding the admin login form

- What to do next - exploring the main configuration content

Create a custom theme

If you want your Developer Portal to have your corporate branding and style, you need to create a custom theme. Themes are composed primarily of Cascading Style Sheets (CSS) files, although they can include much more; see Advanced theme development.

Directly editing the API Connect theme is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed. The way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. A sub-theme inherits the parent theme's resources, and this means that your custom sub-theme CSS file needs to contain only the changes or overrides that you want to make from the default theme. The CSS file can contain as little or as many updates as you like.

For more information, see Creating a sub-theme. You can also follow a tutorial that takes you through how to use the theme generator, customize a theme, and install a custom theme; see Tutorial: Creating a custom theme for the Developer Portal.

While experience with Drupal is beneficial, the only skill you really need to create a sub-theme is CSS.

Changing the page layout

While the theme defines *regions* of the page, such as a footer, or sidebar first, the page layout is defined by configuring which blocks appear in which regions. Blocks are controlled by using the Structure > Block layout options on the Developer Portal administrator dashboard. You can then further limit those blocks to show only on specific pages, or show on all pages except certain ones, to show only for certain roles, or for certain languages, and more.

For information about how to configure the page layout, see Adding and changing the blocks displayed on Developer Portal pages.

Changing the page layout is done entirely in the Developer Portal UI as an Administrator.

Changing the front page

The front page is a special case and does not use the blocks system that is mentioned previously. Instead, you configure it from the Structure > Pages options on the Developer Portal administrator dashboard.

For information about how to configure the front page, see Configuring the front page.

Further options that you might want to consider when designing your front page include:

- Change the banner block content or image: Changing the front page banner block.
- Display featured API Products by creating a featured content block: Changing the front page Featured Content block.
- Create custom blocks with your own HTML content: Adding and changing the blocks displayed on Developer Portal pages.
- Set up the social block to include your organization Twitter feed: Integrating X data into the social block.
- Add a carousel to the front page to display a series of images: Implementing an image carousel.

Again, changing the front page is done entirely in the Developer Portal UI as an Administrator.

Advanced theme development

The DOM structure for specific blocks and parts of the page is controlled by templates. These templates are Twig files that define the actual HTML output for a particular piece of content. You can override a Twig template by copying the original template into the templates folder of your custom sub-theme, and then editing the template to your specification. The Developer Portal then uses your template in preference to the original one. For more information, see Applying a modified content type template.

Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

If you're developing custom templates, knowledge of Twig templating language, Drupal, HTML, and CSS is recommended.

Custom modules

If you need to modify the functional behavior of the Developer Portal, then you can create a custom module to do so. Drupal has an extensible programmatic API that is built around a system of *hooks*. By using these hooks in your custom modules you can change what happens when, for example, forms are displayed, forms are submitted, or make extra variables available to custom templates, and other options.

For more information on writing custom modules, see Extend. You can also follow a tutorial that includes using a custom module, see Tutorial: Adding validation to a field on the sign-up form.

Custom modules are written in PHP, and so Drupal experience and PHP knowledge are very beneficial.

Hiding the admin login form

If your Developer Portal is public facing, you might want to hide the admin registry as an option on the login form. You can do this simply from the administrator dashboard, by clicking Configuration > System > IBM API Developer Portal, and selecting Hide the admin registry on the login form. For more information, see Hiding the admin registry on the login form.

What to do next - exploring the main configuration content

There are an almost infinite number of configuration options available to you on the Developer Portal. Explore the following sections to find out more:

- Appearance - control general appearance elements, and learn how to create a custom theme.
- Content - add, configure, and control content elements, such as meta tags, custom pages, custom content types, and adding content to an API or Product.

- Structure - control how the content in your Portal is displayed, for example configuring the front page, changing block display, and using the Views module.
- Configuration - configure content moderation, perform general configuration tasks, and manage Developer Portal security.
- People - create and customize site configuration roles, control access to sites and to content, and create internal Consumer organizations to allow customization of specific API properties.
- Forums - create and control forums.
- Reports - list of available reports about the general configuration and status of your site.
- Extend - introduction to and instructions for creating custom modules, including information about developing in PHP, and how to use hooks in custom modules.
- Developer Portal tutorials - guided examples taking you through the simple and the more complex areas of configuration.

Appearance

Control and configure the general appearance of your Developer Portal site by setting and configuring a default theme.

To learn how to control the appearance of your Developer Portal site, use the following topic links.

- **Controlling general appearance elements in the Developer Portal**
  You can control general appearance aspects of the Developer Portal.
- **Adding custom JavaScript to a custom theme**
  To increase customization of the Developer Portal, you can add custom JavaScript to custom themes that you install.
- **Applying a modified content type template**
  You can apply a modified content type template to override a default content type template in the Developer Portal. For example, you can override the product content type template to change its layout.
- **Creating a sub-theme**
  Create a custom theme for your Developer Portal site by generating and configuring a sub-theme. A sub-theme is a theme that inherits all the resources of a specified parent theme. You can then override specific resources to configure your required customizations.
- **Installing additional themes**
  You can install additional themes in the Developer Portal.
- **Deleting themes**
  You can delete a theme from your Developer Portal site.

Controlling general appearance elements in the Developer Portal

You can control general appearance aspects of the Developer Portal.

To learn how to control general aspects of the Developer Portal appearance, use the following topic links:

- **Changing the shortcut icon**
  You can change the shortcut icon, or favicon, in the Developer Portal.
- **Changing the site email address**
  You can change the site email address in the Developer Portal.
- **Changing the site logo**
  You can change the site logo for your Developer Portal.
- **Changing the site name or site slogan**
  You can change the site name, or the site slogan, for your Developer Portal.
- **Changing the visibility of site branding**
  You can toggle the visibility of your site branding elements for your Developer Portal.

Changing the shortcut icon

You can change the shortcut icon, or favicon, in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

The shortcut icon, or favicon, is displayed in the address bar or bookmarks of most browsers. Drupal provides a default favicon, the water drop logo, but if you want to make your Developer Portal stand out, you should provide your own.
The default Drupal favicon is 32 pixels high by 32 pixels wide. As a favicon is displayed only in the address bar and favorites (bookmarks) list, any favicon that you create should be similarly small.

Procedure

To change the shortcut icon, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Appearance in the administrator dashboard.
3. Click Settings for the default theme.
4. Click Favicon in the Override Global Settings section.
5. Deselect the Use the favicon supplied by the theme check box.

6. Provide a path to a custom icon on the server by entering it in the Path to custom icon field. Alternatively, you can browse for, and upload, a favicon image under the Upload favicon image subheading.
7. Click Save configuration.
   Note: If the changes don't appear immediately in your browser, clear your browser's cache and reload the page. If you've bookmarked the site, you may need to delete the bookmark and then create it again, so the new favicon is used.

Results

You have changed the favicon for the site.

Changing the site email address

You can change the site email address in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To change the site email address, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. Under the System heading, click Basic site settings.
4. In the SITE DETAILS section, enter the new site email address in the Email address field.
5. Click Save configuration.

Results

You changed the email address for the site.

Changing the site logo

You can change the site logo for your Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To change the site logo, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Appearance in the administrator dashboard.
3. Click Settings for the default theme.
4. Click Logo image in the Override Global Settings section.
5. Deselect the Use the logo supplied by the theme check box.
6. Provide a path to your custom logo by completing the Path to custom logo field. Alternatively, you can browse and upload a logo image under the Upload logo image subheading.
7. Click Save configuration.
   Note: If the changes don't appear immediately in your browser, clear your browser's cache and reload the page.

Results

You changed the site logo for your Developer Portal. How the site logo is used depends on the settings for your theme.

Changing the site name or site slogan

You can change the site name, or the site slogan, for your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

The site name is used to identify the site and can be displayed in browsers. You can also set a site slogan for your theme.

Procedure

To change the site name or site slogan, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration > System > Basic site settings in the administrator dashboard.
3. In the SITE DETAILS section, enter the new site name in the Site name field, or enter a new slogan in the Slogan field.
4. Click Save configuration.

Results

The site name or site slogan is updated. How the site name and the site slogan are used depends on the settings for your site theme. For more information, see Changing the visibility of site branding.

Changing the visibility of site branding

You can toggle the visibility of your site branding elements for your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

The site branding block includes the site logo, name, and slogan. You can choose which of these branding elements you want to be displayed in this block.

Procedure

To change the visibility of your site branding elements, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Block layout in the administrator dashboard.
3. Find the Site branding block in the Navigation section, and click Configure.
4. In the TOGGLE BRANDING ELEMENTS section, use the check boxes to select which branding elements you want to be visible.
5. Click Save block to save your changes.

Results

You changed the visibility of your site branding elements. How these elements are used depends on the settings for your theme.

Adding custom JavaScript to a custom theme

To increase customization of the Developer Portal, you can add custom JavaScript to custom themes that you install.

Procedure

1. Add your JavaScript file to your theme. For example, in a sub-directory named js and the file `myfilename.js`.
2. Add the JavaScript file to your theme, by editing your {custom_theme}.libraries.yml file and adding in the JavaScript file.
   An example of the JavaScript file:

   ```
   myfile:
     js:
       js/myfilename.js: {}
   ```

3. Add the following code to your theme's .info.yml file:

   ```
   libraries:
   - {custom_theme}/myfile
   ```

   Note: You must replace `{custom_theme}` with your theme name. Also, both files must reference `myfile`.
4. Zip up the directory and upload it to your portal site. For more information, see Installing additional themes.

Applying a modified content type template

You can apply a modified content type template to override a default content type template in the Developer Portal. For example, you can override the product content type template to change its layout.

Before you begin

You must have administrator access to complete this task.

You must also have a custom sub-theme, and a copy of an existing template that you want to update. For information about how to create a sub-theme, see Creating a sub-theme.

To obtain a copy of an existing template, you can download a copy from the following locations on GitHub:

- https://github.com/ibm-apiconnect/devportal/tree/master/ibm_apim/templates
- https://github.com/ibm-apiconnect/devportal/tree/master/apic_api/templates
- https://github.com/ibm-apiconnect/devportal/tree/master/apic_app/templates
- https://github.com/ibm-apiconnect/devportal/tree/master/product/templates
- https://github.com/ibm-apiconnect/devportal/tree/master/consumerorg/templates

Note that you must copy the template to your sub-theme, and then modify the content. Do not modify the content that is stored on GitHub.

About this task

You can override the Developer Portal core templates by creating a custom sub-theme, and then adding your modified templates to the sub-theme folder. A sub-theme inherits all of the settings of its parent theme, apart from the settings that are overridden, such as by applying modified templates. After you have applied a modified content types template, these changes take precedence over the default content types template. The Developer Portal templates are formatted as Twig template files. The following levels are examples of where you can use templates:

- html.html.twig - the template for Developer Portal HTML pages
- page.html.twig - the body of Developer Portal HTML page
- node.html.twig - the template for all of the content nodes
- comment.html.twig - the template for all of the comment nodes
- search-result.html.twig - the template for search results
- node--product.html.twig - the template for nodes that use the Product content type
- node--product--teaser.html.twig - the template for the previews for the Product content type

For more information about Twig templates, see Working with Twig templates and Debugging Twig templates on the Drupal documentation website.
Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

Important:
Directly editing any API Connect theme files is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.

Procedure

To override a default content type template:

1. Apply the modifications that you require to the default template that you obtained.
   Warning: Although the ability to override templates is supported, if you override a template they are your responsibility. Templates can be from multiple sources and compatibility with an earlier version is not ensured. You must check the templates in the latest API Connect release to check whether you need to make equivalent changes to your overrides to maintain functional equivalence.
2. Create a directory in your custom sub-theme, and name it templates.
3. Add your modified template to the templates directory.
   Note: The modified template file name must be identical to the original name of the template file that it is overriding.
4. Log in to the Developer Portal UI as an administrator.
5. If the administrator dashboard is not displayed, click Manage to display it.
6. Install your custom sub-theme onto the Developer Portal by clicking Appearance > Install new theme on the administrator dashboard.
7. Click Choose file, select your sub-theme, and then click Install.
8. Click Install newly added theme, find your new theme in the list of Uninstalled themes, and click Install and set as default to set your new custom sub-theme as the default theme for your site.
   Your new theme, which contains the modified template file, is now set as the default theme, and is listed in the Installed themes section.

Results

You successfully modified and applied a template to override a default content type template in the Developer Portal.

Creating a sub-theme

Create a custom theme for your Developer Portal site by generating and configuring a sub-theme. A sub-theme is a theme that inherits all the resources of a specified parent theme. You can then override specific resources to configure your required customizations.

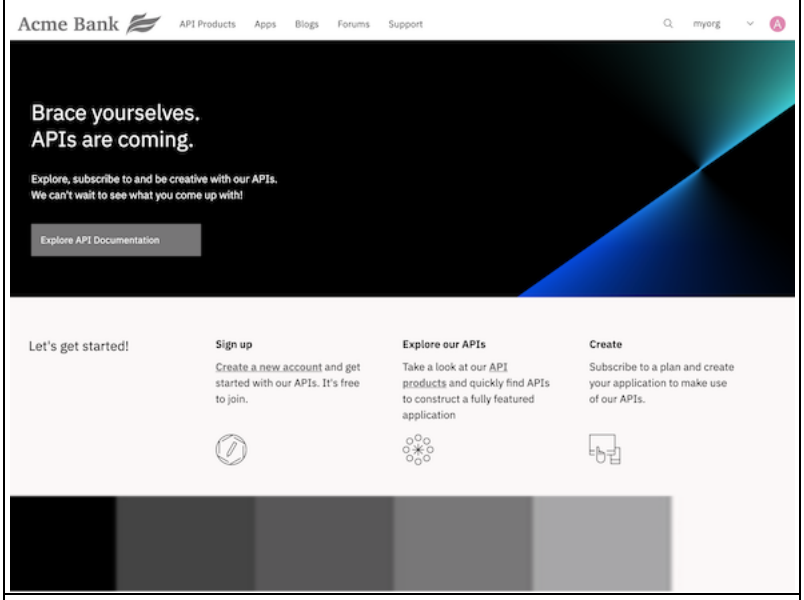
Before you begin

You must have administrator access to complete this task.

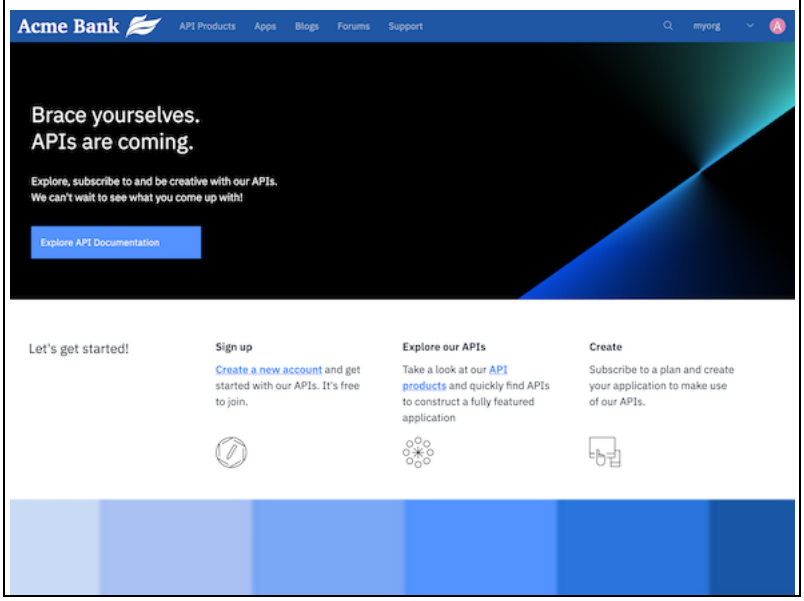
About this task

Directly editing the API Connect theme isn't permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed. So, the way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. You can also create a sub-theme that is based on one of the following color templates:

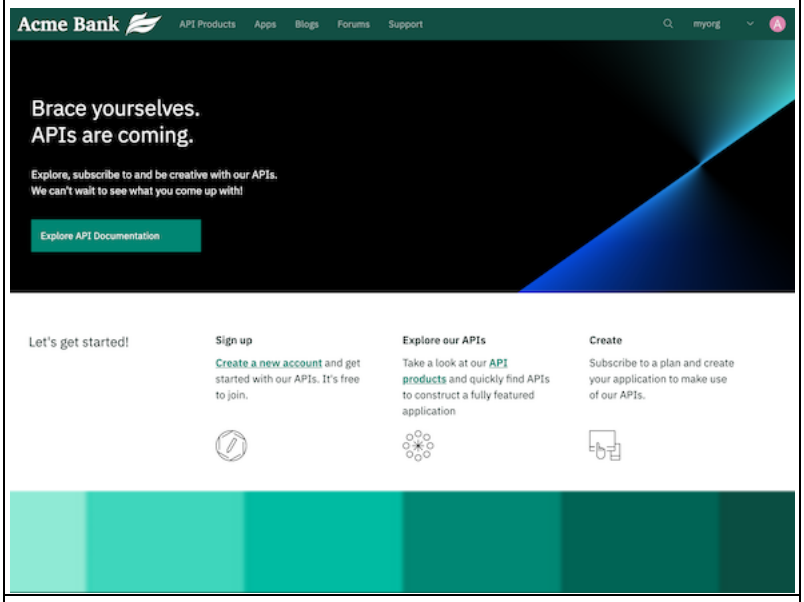
Business Mono



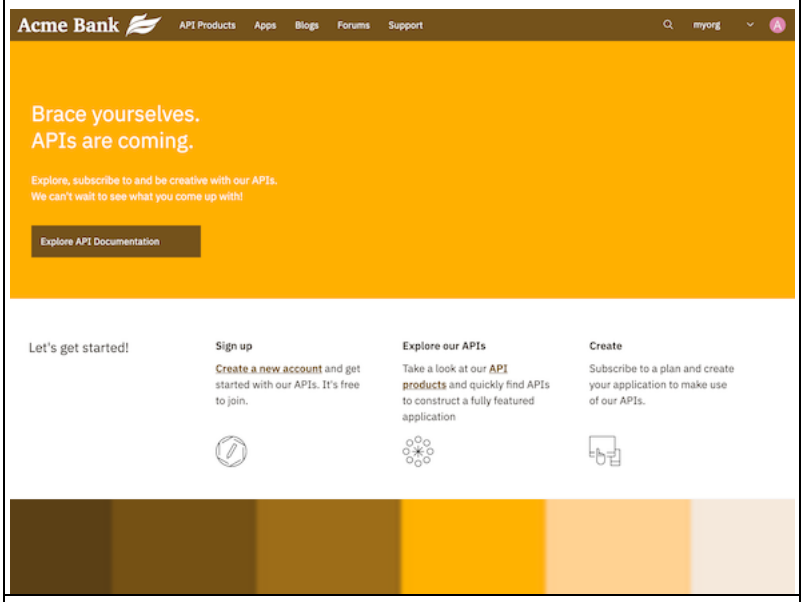
Sapphire Blue



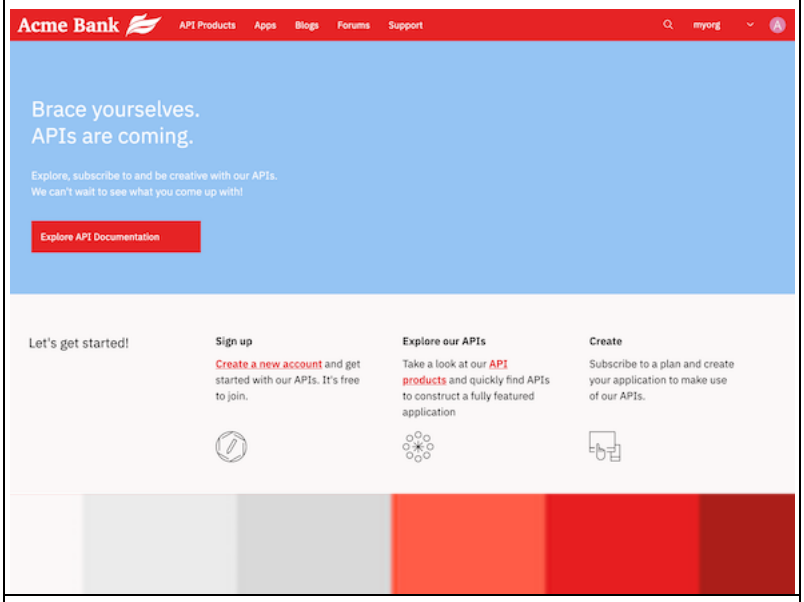
Emerald Green



Golden Brown



Ruby Red



Selecting a color template provides you with a base theme that you can build on more easily to produce your required site branding and styling.
Create a sub-theme by using the theme generator in the Developer Portal UI, customize the appearance as required, and then upload the compressed file back onto the site.

Important:

- You're not permitted to include any IBM® API Connect code within any custom themes that you create. Also, directly editing any API Connect themes on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom themes is supported, IBM API Connect do not provide support in their development or modification.

For a guided example on creating a sub-theme, see Tutorial: Creating a custom theme for the Developer Portal. In this example, you create a sub-theme, and then customize the styling by editing the overrides.css file.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Appearance > Generate sub-theme.
   The Generate sub-theme page is displayed.
3. Enter a Sub-theme name, and select the Sub-theme type that you would like to use. The name can contain only lowercase characters a - z, and numerals 1 - 9.
   - CSS - Cascading Style Sheets; the language that is used to describe how to display the HTML elements in the theme.
   - SCSS - Sass CSS; a superset of CSS. SCSS contains all the features of CSS, but is extended to include the features of Sass as well. However, SCSS must be compiled into CSS before it can be installed and used in the Developer Portal.
4. Select the template to base your sub-theme on:
   - Default - the standard connect_theme that the Developer Portal is based on.
   - Business Mono - a black and white color theme.
   - Sapphire Blue - a blue color theme.
   - Emerald Green - a green color theme.
   - Golden Brown - a brown color theme.
   - Ruby Red - a red color theme.
5. Click Generate.
   Your sub-theme is available to download from the Generate sub-theme page.
6. Click your new sub-theme and save it to your preferred location.
7. Extract the resources from the sub-theme file, and then edit the specific resources to configure the customizations that you require.
   Define your theme by creating metadata in the *your_theme_name*.info.yml file. You can then customize the appearance of your theme by configuring the overrides.css file. For detailed information about how to customize themes, see Theming Drupal.
8. After you have completed your customizations, compress the sub-theme resources back into the .zip file.
   Note that the following file extensions are also supported: tar, tgz, gz, and bz2.
9. Install your new sub-theme onto the Developer Portal by clicking Appearance > Install new theme.
10. Click Choose file, select your sub-theme, and then click Install.
11. Click Install newly added theme, find your new theme in the list of Uninstalled themes, and click Install and set as default to set your new custom sub-theme as the default theme for your site.
    Your new theme is now set as the default theme, and is listed in the Installed themes section.

Results

You created and installed a new sub-theme for your Developer Portal site.

What to do next

You can edit the display settings for your new theme, by clicking Settings next to your theme.

Related information

- Tutorial: Creating a custom theme for the Developer Portal

Installing additional themes

You can install additional themes in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To install additional themes, complete the following steps:

1. Click Appearance in the administrator dashboard.
2. Click + Install new theme.
3. You can enter a path to the theme in the Install from a URL field. Alternatively, you can upload a theme under the Upload a module or theme archive to install heading.
4. Click Install.
5. Click Enable newly added themes.

6. Click Enable and set default for the theme that you want to be displayed in the Developer Portal.
   Your theme is displayed in the Developer Portal when you navigate to the home page.

Deleting themes

You can delete a theme from your Developer Portal site.

Before you begin

You must have administrator access to complete this task.

About this task

If you have a custom theme which is disabled, you can remove it from the Developer Portal. Deleting a theme is useful if you have identified a theme that you do not want to use again.

Procedure

1. Click Appearance from the administrator dashboard.
2. Identify the custom theme that you want to delete, and ensure that it is disabled.
3. Click Delete against the theme.
   The theme is now deleted from your Developer Portal site.

Results

You deleted a disabled theme in the Developer Portal.

Content

Control multiple content entities in the Developer Portal, including the customization and restriction of specific content.

A *content entity* is an item of content data, which can consist of text, HTML markup, images, attached files, and other data that is intended to be displayed to site visitors. Content entities can be defined by the core software or by modules. Content entities are grouped into *entity types*, which have different purposes and are displayed in different ways on the site. Most entity types are also divided into *entity sub-types*, which are divisions within an entity type to allow for smaller variations in how the entities are used and displayed.

The Developer Portal site contains the following content entity types:

- API
- Application
- Article
- Basic page
- Blog post
- Book page
- Consumer Organization
- FAQ
- Forum Topic
- Product

Within content entity items the data is stored in individual *fields*, each of which holds one type of data, such as formatted or plain text, images or other files, or dates. Field types can be defined by the core software or by modules.

Fields can be added by an administrator on content entity sub-types, so that all the entity items of a given entity sub-type have the same collection of fields available. When you create or edit entity items, you are specifying the values for the fields on the entity item.

To learn how to configure the content entities of your Developer Portal site, use the following topic links.

- **Adding content elements in the Developer Portal**
  You can add content elements in the Developer Portal.
- **Configuring and restricting content in the Developer Portal**
  You can configure and restrict certain content elements in the Developer Portal.
- **Turning content on or off in the Developer Portal**
  You can turn certain content elements on or off in the Developer Portal

Adding content elements in the Developer Portal

You can add content elements in the Developer Portal.

Note: You can export and import entities from the Developer Portal UI. When you're editing a content entity type in the UI, you can click Export in the side navigation bar. To create an export entity container and export that entity, including any required embedded entities, click Export entity. Or, if you have an existing export entity container,

you can select the required container, and click Add to the export. You can also view and manage all of your export entities by clicking Content > Content synchronizer > Exports, and all of your import entities by clicking Content > Content synchronizer > Imports.
To learn how to add various content elements in the Developer Portal, use the following topic links:

- **Adding and configuring meta tags**
  You can customize the Developer Portal by adding and configuring meta tags.
- **Adding custom pages**
  You can create custom pages within the Developer Portal. You can add a basic page for static content, or an article for news content.
- **Integrating X data into the social block**
  You can retrieve and integrate data from an X (formerly known as Twitter) account of interest and display it into a social block in a Developer Portal site.
- **Adding content in multiple languages**
  You can create content in multiple languages in the Developer Portal.
- **Adding custom pages to APIs and Products**
  After you create a custom page in the Developer Portal, you can add a link to the new page to any APIs and Products that exist in the Developer Portal.
- **Adding new frequently asked questions**
  You can add new FAQs to the Developer Portal.
- **Applying an image to an API or Product**
  You can apply an image to an API or a Product in the Developer Portal.
- **Attaching a documentation file to APIs**
  You can attach a file to APIs in the Developer Portal, and display them as documentation attachments. You can attach multiple files to APIs.
- **Configuring blogs**
  You can post new blog entries in the Developer Portal, and configure them to suit your requirements. You can also turn blogs off.
- **Configuring the taxonomy menu block**
  To display your tag hierarchy in the Developer Portal, you must configure the taxonomy menu block.
- **Customizing the URL alias for a specific API or Product page**
  You can customize the URL of an API or Product page from the default alias that they are assigned.
- **Embedding multimedia in site content**
  You can embed multimedia elements in site content in the Developer Portal.
- **Linking from one piece of site content to another**
  You can link from one piece of site content to another in the Developer Portal.
- **Linking to social media sites**
  You can link to social media sites from the Developer Portal. You can configure where and how these links are displayed and which sites you want to link to.
- **Managing tags in the Developer Portal**
  These instructions show you how to add new tags to your taxonomy, and how to manage the tag hierarchy in the Developer Portal. You can use tags to classify your Developer Portal content.
- **Posting a poll**
  You can post a poll in the Developer Portal.
- **Adding images to site content**
  You can add .jpg, .png and .gif images to your site content in the Developer Portal.

Adding and configuring meta tags

You can customize the Developer Portal by adding and configuring meta tags.

Before you begin

You must have administrator access to complete this task.

About this task

Meta tags provide preview information that is displayed for a link to a website. The preview information can include images, descriptions, tags, and links. Meta tags are used when you link to external website, for example Facebook, Twitter, and Slack.

You can see which entities have the meta tag form available on their respective edit pages by default, and can be configured, Configuration > Search and metadata > Metatags > Settings:

- CONTENT
  - API
  - APPLICATION
  - ARTICLE
  - BLOG POST
  - BOOK PAGE
  - CONSUMER ORGANIZATION
  - FAQ
  - FORUM TOPIC
  - BASIC PAGE
  - PRODUCT
- TAXONOMY TERM
  - FORUMS
  - TAGS
- USER
  - USER

Procedure

1. In the administrator dashboard, click Extend > Update, then click List.

2. Enter `meta tag` in the Filter list field.
3. Select the check boxes for the Meta tags you want to enable, click Enable.

- **Disabling external search engine indexing**
  If you do not want your Developer Portal site content to be indexed by external search engines, then you can disable the indexing.

Disabling external search engine indexing

If you do not want your Developer Portal site content to be indexed by external search engines, then you can disable the indexing.

Before you begin

You must have administrator or content author access to complete this task.

About this task

By disabling the external search indexing, you can increase the confidentiality of your content and restrict its exposure.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration.
3. In the SEARCH AND METADATA section, click Metatag.
4. Click Edit for the GLOBAL type.
5. Expand the ADVANCED section and, in the Robots section, select Prevents search engines from indexing this page.
6. Click Save.

Results

You have disabled external search engine indexing.

Adding custom pages

You can create custom pages within the Developer Portal. You can add a basic page for static content, or an article for news content.

Before you begin

You must have administrator or content author access to complete this task.

About this task

Use basic pages for your static content, such as an "About us" page. Typically, a basic page is used for static content that can be linked into the main navigation bar, although this is not a requirement. Use articles for time-sensitive content, like news, press releases, or blog posts.

Procedure

To add custom pages, complete the following steps.

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Content.
4. Click +Add content, then click either Basic page or Article, depending on the type of content you want to display.
5. Enter a page title, select the language, and enter the content.
6. Specify the remaining page options as required.
   If you want your new page to be linked to from a menu, you must complete the Menu settings section.
7. Optional: Click Edit summary and specify a summary page description.
8. Click Save to publish the new page immediately. If you don't want to publish the new page immediately, ensure that you deselect Published before saving.

Results

You added a new custom page to the Developer Portal. You can edit the page after it is created by clicking Edit for the custom page.

Integrating X data into the social block

You can retrieve and integrate data from an X (formerly known as Twitter) account of interest and display it into a social block in a Developer Portal site.

Before you begin

You must have administrator access to complete this task.

You must have an X App for the X account of interest. The X App contains information that is used to configure the social block. For information on creating X Apps, see https://apps.twitter.com/.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration.
3. In the WEB SERVICES section, click IBM Social Block settings.
4. Using the information from your X App, enter the Consumer Key, Consumer Secret, Access Token, and Token Secret in the corresponding fields.
5. Click Save configuration.
   X data from an X account of interest has been integrated with the Developer Portal, and is displayed on the social block of the site.

- **Editing the social block**
  After you have integrated Twitter data into the Developer Portal social block, you can edit the content, appearance, and functionality of the social block.

Editing the social block

After you have integrated Twitter data into the Developer Portal social block, you can edit the content, appearance, and functionality of the social block.

Before you begin

You must have administrator access to complete this task.

You must have Twitter data integrated into the Developer Portal social block. For more information, see Integrating X data into the social block.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Structure > Pages.
3. Click Edit next to the Welcome page (or next to the page that contains the social block that you want to edit).
4. From the Variants navigation tree, select Panels > Content, and click Edit next to the Social Block.
   The Edit block window is displayed.
5. Edit the social block by configuring any of the following options:

   Title
   : Update the title as required.

   Number of tiles to display
   : Configure the number of tiles that you want to display on the social block by entering a number in the Number of tiles to display field. The default number of tiles that are displayed on the social block is 9, and there is a limit of 100 tiles.

   FORUM
   : Use the check boxes to specify the forum topics that you want to display.

   Get tweets from
   : From the Get tweets from drop-down list, select whether you want to display Twitter data from users or a search term.

   Twitter search parameter
   : Enter the Twitter search parameter.

   Types of tweets to display
   : Select the types of tweets that you want to display from the drop-down list. You can select to display tweets, or tweets and replies.

6. After you have configured the social block, click Update block.
   Your social block is configured, and the configurations are implemented when you return to the home page.

Adding content in multiple languages

You can create content in multiple languages in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

About this task

You can provide custom pages in multiple languages to your APIs and Products to improve your customer experience and understanding. By default, multiple languages are enabled in the Developer Portal.

Note: In API Designer, translations for APIs and products must be done within a document. For more information, see Using `x-ibm-languages` to create multilingual API and Product documentation.

Procedure

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Content.
4. If you are creating a new page to be translated, click +Add content.
    a. Click Basic page, then select type, for example **`Basic page`**.
    b. Enter a title for your custom page in the Title field.
    c. Enter your information in the Body field.
    d. Click Save.
    e. Click Translate.
    f. Click Add for the language you want to translate.
    g. Provide the language content.
    h. Click Save (this translation).
5. If you are translating an existing page, find the page that you want to update in the list.
    a. Click Translate from the drop-down list for the page.
    b. Click Add for the language you want to translate.
    c. Provide the language content.
    d. Click Save (this translation).

Results

You successfully created a translated version of your content.

Adding custom pages to APIs and Products

After you create a custom page in the Developer Portal, you can add a link to the new page to any APIs and Products that exist in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

You must have an API or Product to add the custom page to.

About this task

You can add custom pages to your APIs and Products to provide further information on their use and value.

For example, you can create a custom documentation page on the implementation of an API in a specific circumstance, and add this custom documentation page to the API.

Procedure

1. Log in to the Developer Portal as an administrator or content author.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Content, then click +Add content.
4. Click Basic page.
   The Create Basic page window is displayed.
5. Enter a title for your custom page in the Title field.
6. Enter your information of interest in the Body field.
7. If you want to add the custom page to a specific Product, enter the Product name in the Link to one or more specific Products field. As you type into this field, a list of available Products is displayed for you to select. You can add more Products by clicking Add another item. If you want to add the custom page to all Products, select the Link to all Products check box.
8. If you want to add the custom page to a specific API, select the APIs tab, and then enter the API name in the Link to one or more specific APIs field. As you type into this field, a list of available APIs is displayed for you to select. You can add more APIs by clicking Add another item. If you want to add the custom page to all APIs, select the Link to all APIs check box.
9. Optional: Configure the additional settings of the custom page to your specification.
10. Click Save.

Results

Your custom page is created and linked to the APIs or Products that you have specified. You can edit the page after it is created, and update the APIs and Products that it is linked to, by clicking Edit for the custom page.

Adding new frequently asked questions

You can add new FAQs to the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

About this task

To add new FAQs, you create a new FAQ node and add questions and answers to that node. Each FAQ node represents an FAQ topic, and each topic can contain multiple questions and answers.

Procedure

To add new FAQs, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Content in the administrator dashboard.
4. Click +Add content.
5. Click FAQ.
   The Create FAQ page displays.
6. Provide a Title for the new FAQ topic.
7. Enter the question in the Question field, and enter the answer in the Answer field.
8. Optional: If you have more content for this FAQ topic, click Add another item and complete the Question and Answer fields. Repeat this step as required.
9. Click Save to save your new FAQ topic.
   Your new FAQ topic is displayed in the Support section of the Developer Portal.

Results

You successfully added new FAQs to the Developer Portal.

What to do next

You can edit the current content of FAQs by clicking Content in the administrator dashboard, finding the FAQ content that you want to edit, and clicking Edit against that FAQ.

Applying an image to an API or Product

You can apply an image to an API or a Product in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To add an image to an API or Product, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click Edit next to the API or Product that you want to edit.
4. Under the Image heading, click Choose File to select an image.
5. Click Save and keep published, to publish the updated content,or select Save and unpublish if you want to publish the updated content later.
   You have added a new picture to the selected API or Product.

Attaching a documentation file to APIs

You can attach a file to APIs in the Developer Portal, and display them as documentation attachments. You can attach multiple files to APIs.

Before you begin

You must have administrator or content author access to complete this task.

About this task

You can attach only one file at a time. The default file upload size limit applies to the files that you want to attach, and you can attach up to 10 files to an API by default.
Note: You can change the default limit for the number of files that you want to attach.
An administrator can configure the types of file that are uploaded, and the number of files that a user can upload.

The following file types can be uploaded:

- txt doc pdf
- xls ppt pptx
- docx xlsx rtf

- odt ods odp md json
- yaml tgz tar zip

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click Edit next to the API you want to attach the file to.
4. In the Documentation section, click Browse and identify the file that you want to upload, then click Open.
5. Click Upload, then click Save.
6. Optional: Clear the Display checkbox for the file if you want to upload and attach the file to the API, but not show it in the Developer Portal.
7. Optional: Enter a description for your file in the Description field. This description can be used as the label of the link to the file.
8. Optional: If you uploaded multiple files, you can use the drag handle next to the file name to rearrange them.

Results

You successfully attached a documentation file to your API.

Configuring blogs

You can post new blog entries in the Developer Portal, and configure them to suit your requirements. You can also turn blogs off.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To post a new blog entry, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Content in the administrator dashboard.
- Click Add content.
- Click Blog post.
- Fill in the Title and Body fields.
- Click Save and keep published, to publish the updated content,or select Save and unpublish if you want to publish the updated content later.
  You have posted a new blog entry.

To configure a blog post, complete the following steps:

- If the administrator dashboard is not displayed, click Manage to display it.
- Click Content in the administrator dashboard.
- Click Edit next to the blog post that you want to configure.
- You can configure the selected blog from this view.
- Click Save and keep published, to publish the updated content,or select Save and unpublish if you want to publish the updated content later.
  You have configured the selected blog post.

Configuring the taxonomy menu block

To display your tag hierarchy in the Developer Portal, you must configure the taxonomy menu block.

Before you begin

You must have administrator or content author access to complete this task.

About this task

Along with controlling how your tag hierarchy is displayed through configuring your taxonomy menu block, you can configure options for the taxonomy tree. The following options can be configured for your taxonomy menu block:

- Vocabulary
- Parent option
- Depth
- Node options and content
- Region settings
- Visibility settings

Important: The taxonomy block will appear after configuration only if there are tags for it to display.

Procedure

1. In the Administrator dashboard, click Structure > Blocks.
2. Click configure for the Taxonomy Menu Block (Tags) block.
3. Configure the options that are available for your taxonomy menu block.
4. When you have configured your taxonomy menu block, click Save block.
   You have configured your taxonomy menu block, and your tag hierarchy is displayed based on your configurations.

Customizing the URL alias for a specific API or Product page

You can customize the URL of an API or Product page from the default alias that they are assigned.

Before you begin

You must have administrator or content author access to complete this task.

About this task

Creating a custom URL, or path, alias is useful if you want to make the URL relevant to the API or Product page.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Content.
3. Identify the API or Product page that you want to customize the URL alias for, and click Edit.
4. In the Path Alias text field, enter the custom URL alias that you want to assign your API or Product page.
   Note: All Path Alias must be unique, and can contain only the following characters: [A-Za-z0-9_-.].
5. Click Save to publish the updated content.

Results

You have applied a custom URL alias for your specified API or Product page, and it is displayed in the URL of your browser when the API or Product page is selected. Depending on whether it is added to an API or a Product page, the application of the Path Alias differs. For example, if you add `my_new_path_alias`, you will have:

- `<my_site_url>/api/my_new_path_alias` for an API
- `<my_site_url>/product/my_new_path_alias` for a Product

Embedding multimedia in site content

You can embed multimedia elements in site content in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To embed multimedia in site content, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Under the TITLE heading, click the title of the required content.
4. Click the Edit tab.
5. Use the Insert Media Embed icon in the in the WYSIWYG editor to embed multimedia in the content.
   Note: The Insert Media Embed icon is available only if you have selected Full HTML mode.
6. Click Save and keep published to publish the new page immediately, or select Save and unpublish if you want to publish the page later.
   You have embedded multimedia in the selected site content.

Linking from one piece of site content to another

You can link from one piece of site content to another in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To link from one piece of site content to another, complete the following steps:

1. Click Content in the administrator dashboard.
2. Under the TITLE heading, click the title of the required content.
3. Select the Edit tab.
4. Use the Link to content icon in the WYSIWYG editor.
5. Click Save.
   You have linked the selected site content to another site content element.

Linking to social media sites

You can link to social media sites from the Developer Portal. You can configure where and how these links are displayed and which sites you want to link to.

Before you begin

You must have administrator access to complete this task.

Procedure

1. Click Structure in the administrator dashboard.
2. Click Blocks.
3. Under the Disabled heading, locate the Follow site block.
4. Select the location for your social media block from the drop-down list for Follow Site in the REGION column.
   The Follow Site block will appear under the heading that corresponds to the region you selected in the REGION drop-down list.
5. Required: Click Save blocks before you begin configuring your new block.
6. Click configure for the Follow Site block.
7. In the Block title field, type the name of your block.
8. Under the Default block title heading, select one of the following leading text options for your block using the radio buttons for each option:
   - "Follow *your.portal.site* on"
   - "Follow me on"
   - "Follow us on"
9. Use the User pages check box to decide whether you want your block to display on your user's profile pages.
10. Use the Alignment and Icon Style drop-down lists to adjust the appearance of your block.
11. Click Save block.
12. Click Configuration in the administrator dashboard.
13. Under the WEB SERVICES heading, click Follow.
14. Type the URL under the URL heading for each of the social media sites you want your users to access. The following is an example of a link to the @ibmapiconnect twitter handle:

    ```
    https://twitter.com/ibmapiconnect
    ```

15. Click Submit.

Managing tags in the Developer Portal

These instructions show you how to add new tags to your taxonomy, and how to manage the tag hierarchy in the Developer Portal. You can use tags to classify your Developer Portal content.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To manage your tags, complete the following steps

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Taxonomy > Tags in the administrator dashboard.
   The List window for Tags is displayed.
3. You can add new terms, delete, and reorganize terms from this view.
   a. To add a new term, click Add term, define the new term, then click Save.
   b. To delete a term, select Delete from the OPERATIONS menu next to the term that you want to delete. Then click Delete to confirm.
   c. To reorganize terms, ensure that the row weights are hidden by clicking Hide row weights if necessary, then use the drag-and-drop handles to manage the tag hierarchy. Terms can be dragged to organize a parent grouping hierarchy. Then click Save.

- **Tagging APIs based on their lifecycle phase**
  In the Developer Portal, you can have APIs automatically tagged with their lifecycle phase; realized, identified, or specified.

Related tasks

- Editing tags for a specific item

Tagging APIs based on their lifecycle phase

In the Developer Portal, you can have APIs automatically tagged with their lifecycle phase; realized, identified, or specified.

Before you begin

You must have administrator or content author access to complete this task.

About this task

APIs are tagged in the API Manager with a lifecycle phase as follows:

- Realized (default) - The API is in the implementation phase.
- Identified - The API is in the early conceptual phase and is neither fully designed nor implemented.
- Specified - The API has been fully designed and passed an internal milestone, but has not yet been implemented.

You can configure the Developer Portal to tag APIs with their lifecycle phase automatically.

Procedure

Tagging APIs with their phase is disabled by default. To enable the function:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration.
3. In the SYSTEM section, click IBM API Connect.
4. Select Automatically tag APIs with their phase, and then click Save configuration.

What to do next

You can manage the tag hierarchy of your APIs. For more information, see Managing tags in the Developer Portal.

Posting a poll

You can post a poll in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To post a poll, complete the following steps:

1. Click Content in the administrator dashboard.
2. Click Add content.
3. Click Poll.
4. Enter the poll question in the Poll question field.
5. Under the Choice heading, add answers to the available fields as choices.
6. From the Poll duration drop-down menu, select a time limit for the poll.
7. Click Save.
   You have posted a poll.

What to do next

See Editing sample content to learn how to add a links to your poll to sample content.

Adding images to site content

You can add .jpg, .png and .gif images to your site content in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To upload images for use in site content, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Under the TITLE heading, click the title of the required content.
4. Click the EDIT tab.
5. Use the Image icon in the in the WYSIWYG editor to add an image.
6. Click Save and keep published to publish the new page immediately, or select Save and unpublish if you want to publish the page later.

Configuring and restricting content in the Developer Portal

You can configure and restrict certain content elements in the Developer Portal.

To learn how to configure and restrict content in the Developer Portal, use the following topic links:

- **Creating content types**
  Content types define default fields for editors to add content on your Developer Portal site and are the building blocks for authoring content. You can control which content types are available
- **Configuring documentation upload limitations**
  You can attach documentation to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded documentation.
- **Configuring image upload limitations**
  You can attach an image file to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded image files.
- **Configuring the appearance of ratings in the Developer Portal**
  You can configure the appearance of ratings in the Developer Portal.
- **Customizing the privacy policy statement**
  You can define and customize a privacy policy statement that sets out the privacy conditions of your Developer Portal site. The privacy policy can be linked to from the cookie compliance banner.
- **Customizing the terms of use statement**
  You can define and customize a terms of use statement that sets out the terms and conditions that your users must accept to use your Developer Portal site.
- **Editing a site comment**
  You can edit site comments in the Developer Portal.
- **Editing sample content**
  You can edit sample content, such as basic pages, in the Developer Portal.
- **Editing tags for a specific item**
  You can edit tags for a specific item in the Developer Portal.

Creating content types

Content types define default fields for editors to add content on your Developer Portal site and are the building blocks for authoring content. You can control which content types are available

Before you begin

You must have administrator access to complete this task.

About this task

After you create your content type, it is added to the following list of default content types that are present in the Developer Portal:

- API
- Application
- Article
- Basic page
- Blog post
- Book page
- Consumer organization
- Forum topic
- Product

The content type that you create, and the default content types, can be edited and configured further to your specifications.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. Click +Add content type.
5. Enter a name for your content type in the Name field.
6. Optional: Enter the text that you want to appear on the Add new content page, in the Description field.
7. Enter a title field label for your content type in the Title field label field.
8. Optional: Configure any of the following options for your content type:

- Submission form settings
- Publishing options
- Language settings
- Display settings
- Menu settings

9. After you have configured your content type to your specifications, click Save and manage fields.
   You have created a new content type.

What to do next

You can add fields to your new content type; see Adding fields to content types.

- **Adding fields to content types**
  Fields can be added to content types in the Developer Portal for customization, and to add functionality.

Adding fields to content types

Fields can be added to content types in the Developer Portal for customization, and to add functionality.

Before you begin

You must have administrator access to complete this task.

About this task

You can specify the type of data that a field can store, and the form element to edit the data with. The type of form element that is available is dependent on the type of data that your field stores.

Note: You can specify the number of values that a field can store, including an unlimited amount. You can configure the value after the field is created. Reducing the number of values that can be stored after the field is created might lead to data loss.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types, and identify the content type that you want to add a field to.
4. Click Manage fields for the content type, then click Add field.
5. Select a field type or select an existing field to reuse, then enter a label for your field in the Label field.
   The label that you specify for your field is displayed in the UI.
6. Optional: If you want to edit the machine name that is automatically created based on your label, click Edit and enter the new name.
   You cannot change the machine name after the field is created.
7. Click Save and continue.
8. Modify the field type specific settings as required, then click Save field settings.
   You have added a field to your content type.

Configuring documentation upload limitations

You can attach documentation to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded documentation.

Before you begin

You must have administrator access to complete this task.

About this task

The set of configurable limitations that you can place on uploading documentation is different to the set of limitations that you can place on uploading images.

The file size limit that is applied when you upload documentation is also different to the limit that is applied when you upload images.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structurein the administrator dashboard.
3. Click Content types.
4. For the API content type, click Manage fields.
5. For the Documentation field, click Edit.
6. Configure the type of files that are allowed to be uploaded in the Allowed file extensions field.
7. Enter the file size upload limit in the Maximum upload size field.
   The default size is 10MB, while the maximum upload size possible is 64MB.
8. Click Save settings.

You have configured the documentation upload limitations.

Configuring image upload limitations

You can attach an image file to an API by uploading it in the Developer Portal. You can configure the API content type to place limitations on the uploaded image files.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structurein the administrator dashboard.
3. Click Content types.
4. For the API content type, click Manage fields.
5. For the Image field, click Edit.
6. Configure the type of image files that are allowed to be uploaded in the Allowed file extensions field.
7. Specify the maximum and minimum allowed image sizes in the Maximum image resolution and Minimum image resolution fields.
8. Enter the file size upload limit in the Maximum upload size field.
   The default size is 2MB, while the maximum upload size possible is 64MB.
9. Click Save settings.
   You have configured the image upload limitations.

Configuring the appearance of ratings in the Developer Portal

You can configure the appearance of ratings in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can configure the appearance of ratings for only the following content types in the Developer Portal:

- APIs
- Applications
- Products

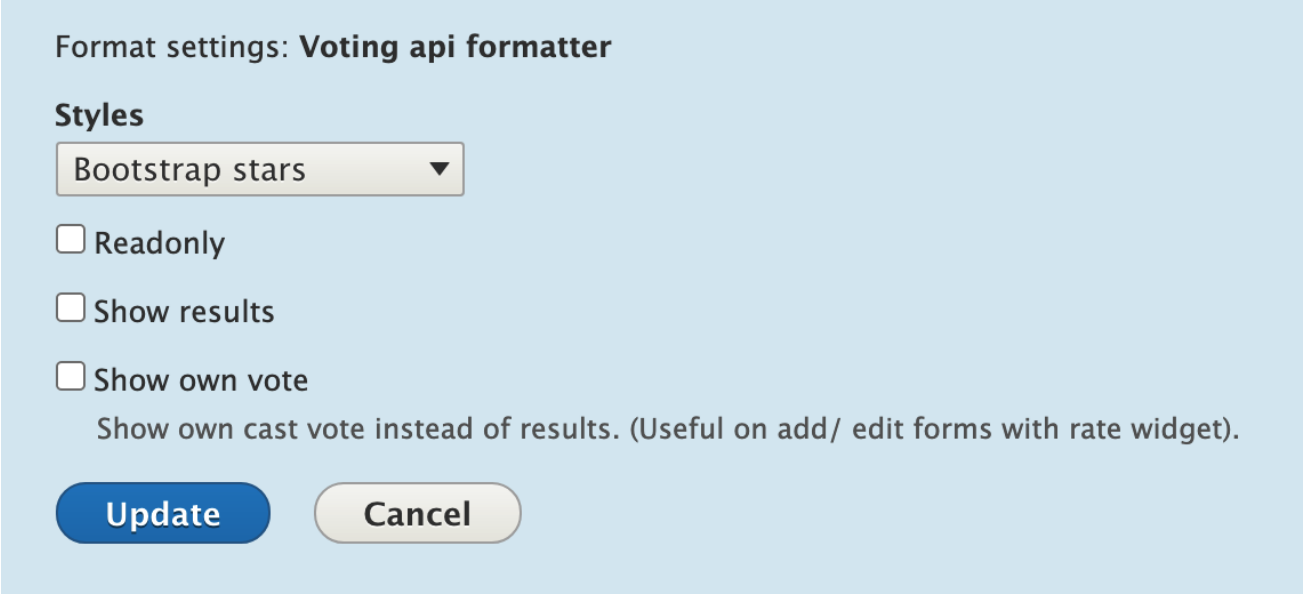
Note: The Applications and Products content types rating functionality is set to hidden by default.

Procedure

These following steps describe how to customize the rating on a selected API, Application, or Product. To ensure that the configured rating is also displayed in the list view of APIs, Applications, or Products, then you must repeat these steps on the Teaser tab in the Manage Display window:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. Select Manage display from the list in OPERATIONS column, next to the required content type.

5. Click the Settings icon , then select the style you require, then click Update.



6. Click Save. You have configured the appearance of the ratings feature for the required content type.

Customizing the privacy policy statement

You can define and customize a privacy policy statement that sets out the privacy conditions of your Developer Portal site. The privacy policy can be linked to from the cookie compliance banner.

Before you begin

You must have administrator access to complete this task.

Procedure

To define or customize a privacy policy, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Content, then enter `privacy` into the title search filter, and click Filter.
   The Privacy Policy content item is displayed in the list of content.
4. Click Edit next to the Privacy Policy content item.
5. Update the content as required. For example, you can complete the following tasks:
   a. Edit the text of the privacy policy statement.
   b. Set the display settings.
   c. Link the content to Products and APIs.
   d. Enter an explanation of the changes that have been made to the privacy policy since the last version.
6. Click Save to save your changes.

Results

The privacy policy statement is successfully updated.

What to do next

You can set your cookie compliance banner to link to your updated privacy policy. For more information, see Enabling a cookie compliance banner.

Related tasks

- Enabling a cookie compliance banner
- Customizing the terms of use statement

Customizing the terms of use statement

You can define and customize a terms of use statement that sets out the terms and conditions that your users must accept to use your Developer Portal site.

Before you begin

You must have administrator access to complete this task.

Procedure

To define or customize a terms of use statement, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Content, then enter `terms` into the title search filter, and click Filter.
   The Terms of use content item is displayed in the list of content.
4. Click Edit next to the Terms of use content item.
5. Update the content as required. For example, you can complete the following tasks:
   a. Edit the text of the terms of use statement.
   b. Set the display settings.
   c. Link the content to Products and APIs.
   d. Enter an explanation of the changes that have been made to the terms of use since the last version.
6. Click Save to save your changes.

Results

The terms of use statement is successfully updated.

What to do next

You can force new users to accept the terms and conditions on the Developer Portal. For more information, see Forcing new users to accept terms and conditions.

Related tasks

- Enabling a cookie compliance banner
- Customizing the privacy policy statement

Editing a site comment

You can edit site comments in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To edit a site comment, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click the Comments tab.
4. In the OPERATIONS column, click Edit alongside the comment that you want to edit.
5. Edit the content of the comment as required.
6. Click Save.
   You have edited the selected comment.

Editing sample content

You can edit sample content, such as basic pages, in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To edit sample content, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. In the OPERATIONS column, click Edit alongside the content item that you want to edit.
4. Edit the sample content of the selected element as required.

5. Click Save and keep published, to publish the updated content,or select Save and unpublish if you want to publish the updated content later.

Editing tags for a specific item

You can edit tags for a specific item in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

You must have some tags that are available to apply to your content; see Managing tags in the Developer Portal.

Procedure

To edit tags for a specific item, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. In the list of content items, click Edit alongside the content item that you want to update.
4. In the Tags section, select the required tag.
5. To add another tag, click Add another item, then select the required tag.
6. If you want to publish the updated content, ensure the Published check box is selected. Deselect the check box if you want to publish the updated content later.
7. Click Save.

Related tasks

- Managing tags in the Developer Portal

Turning content on or off in the Developer Portal

You can turn certain content elements on or off in the Developer Portal

To learn how to turn content on or off in the Developer Portal, use the following topic links:

- **Deleting a site comment**
  You can delete site comments in the Developer Portal.
- **Deleting blocks**
  You can delete blocks in the Developer Portal.
- **Disabling blogs**
  You can disable blogs on your Developer Portal.
- **Deleting forums from the front page**
  You can delete the help information on forums from the front page of your Developer Portal.
- **Turning comments for specific content types on or off in the Developer Portal**
  You can turn the comments for specific content types on or off in the Developer Portal.
- **Turning comments off for an individual item**
  You can turn off comments for individual items within a content type.
- **Turning off ratings for specific content types in the Developer Portal**
  You can turn off ratings for specific content types in the Developer Portal by hiding the ratings field.
- **Turning the ability to tag specific content types off in the Developer Portal**
  You can turn the ability to tag specific content types on or off in the Developer Portal.

Deleting a site comment

You can delete site comments in the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

Procedure

To delete a site comment, complete the following steps

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click the Comments tab.
4. From the UPDATE OPTIONS list, select Delete the selected comments.
5. Select the comments that you want to delete, then click Update.
   The confirmation screen displays.
6. Click Delete comments.

You have deleted the selected comments.

Deleting blocks

You can delete blocks in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Pages.
4. In the OPERATIONS column, find the welcome page row and select Edit from the drop-down list.
5. Expand Panels, then click Content.
6. In the OPERATIONS column, find the block that you want to delete and select Delete for the drop-down list
7. Click Delete to confirm, then click Update and save.

Disabling blogs

You can disable blogs on your Developer Portal.

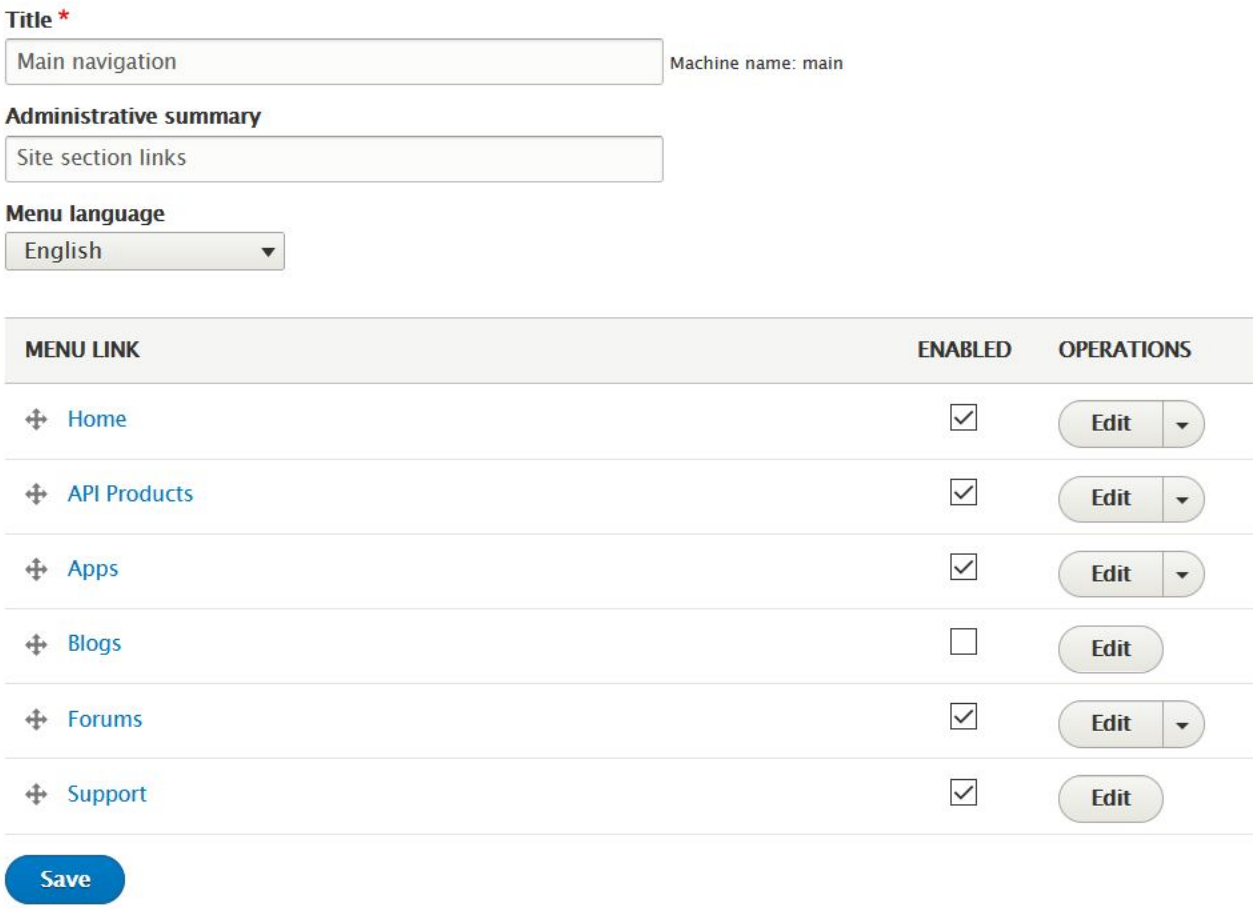
Before you begin

You must have administrator access to complete this task.

Before any changes are made, **`Blogs`** show as an option on the front page like this:

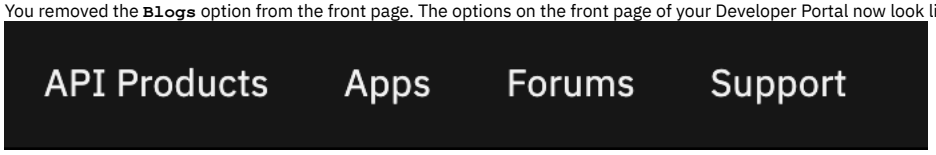
Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Menus from the administrator dashboard.
3. In the OPERATIONS column, click Edit menu for **`Main navigation`**.
4. Clear Enabled for the **`Blogs Menu link`**, then click Save.



Results

You removed the **`Blogs`** option from the front page. The options on the front page of your Developer Portal now look like this:



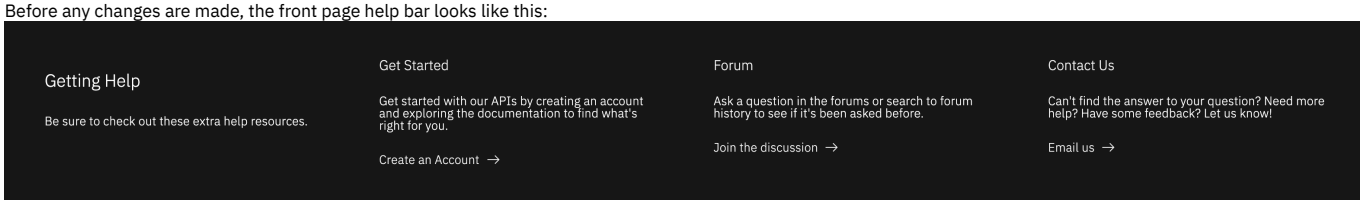
Deleting forums from the front page

You can delete the help information on forums from the front page of your Developer Portal.

Before you begin

You must have administrator access to complete this task.

Before any changes are made, the front page help bar looks like this:



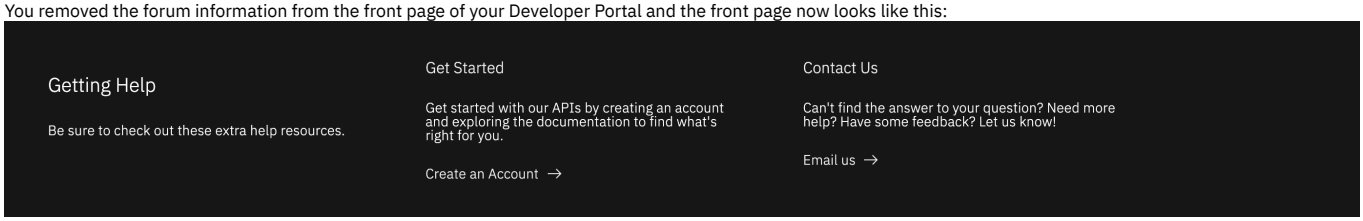
Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Block layout > Custom block library.
4. In Block description, enter `Get help`, then click Apply.

5. In the OPERATIONS column, click Edit for **`Get Help`**.
6. In the **`Body`** section, delete the forum information, then click Save.

Results

You removed the forum information from the front page of your Developer Portal and the front page now looks like this:



Turning comments for specific content types on or off in the Developer Portal

You can turn the comments for specific content types on or off in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To turn comments for specific content types on or off in the Developer Portal, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. In the OPERATIONS column, click Manage fields for the required content type.
5. In the OPERATIONS column, click Edit for the Comments field.
   Note:
   - If there is no Comments field, you can enable commenting for the selected content type by adding one as follows:
     - Click Add field.
     - From the Add a new field list, select Comments.
     - Enter a field label, then click Save and continue.
     - Select the required comment type, then click Save field settings.
   - You can completely disable commenting for the selected content type, and permanently remove all current comments, by selecting Delete.
6. Select the required option in the DEFAULT VALUE section.
   a. To allow commenting, select Open.
   b. To keep existing comments but disallow further commenting, select Closed.
   c. To hide all comments, select Hidden.
7. When done, click Save settings.

Turning comments off for an individual item

You can turn off comments for individual items within a content type.

Before you begin

Users with edit permission for the selected content type can perform this task.

Procedure

To turn off comments for a single item, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. In the OPERATIONS column, click Edit for the required content item.
4. Click Comment settings in the side pane.
5. Select Closed to turn comments off for the selected item.

6. Click Save.

Turning off ratings for specific content types in the Developer Portal

You can turn off ratings for specific content types in the Developer Portal by hiding the ratings field.

Before you begin

You must have administrator access to complete this task.

Procedure

To turn off ratings for specific content types in the Developer Portal, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Content types.
4. In the OPERATIONS column for the required content type, select Manage fields from the drop-down list.
5. Click on the Manage form display tab. In the FIELD list, select Rating and drag and drop it into the Disabled section, then click Save.
6. Click on the Manage display tab. Content items are displayed by using different view modes, such as Default, Card, and Teaser, and these modes are shown at the beginning of the Manage display page. For each view mode that is applicable to the content type you're editing, click the mode, and in the FIELD list select Rating, and drag and drop it into the Disabled section.
7. Click Save.

Turning the ability to tag specific content types off in the Developer Portal

You can turn the ability to tag specific content types on or off in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To turn the ability to tag specific content types on or off in the Developer Portal, complete the following steps:

1. Click Structure in the administrator dashboard.
2. Click Content types.
3. Click Manage fields next to the required content type.
4. From the Tags field, click Edit.
5. Click Delete, then confirm the Delete as the action cannot be undone.
   You have turned the ability to tag specific content types off.

Structure

Control the layout of the Developer Portal, such as configuring the front page, adding and changing blocks, and displaying Products and APIs in categories.

To learn how to control the layout of the content on your Developer Portal site, use the following topic links.

- **Configuring the front page**
  You can configure the Developer Portal front page.
- **Adding a menu**
  You can add a menu to the Developer Portal, and define the items that are included in the menu.
- **Adding and changing the blocks displayed on Developer Portal pages**
  You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages.
- **Changing the front page banner block**
  You can change the front page banner content and image that is displayed on the home page when a user first logs in to the Developer Portal.
- **Changing the front page Featured Content block**
  You can display featured Products or APIs on the front page of your Developer Portal by configuring the Featured Content block.
- **Changing the items in the main menu**
  You can change the items in the main menu that is displayed on all pages in the Developer Portal.
- **Changing the ratings display**
  You can change the appearance of the content ratings in the Developer Portal.
- **Displaying APIs and Products in categories**
  You can create taxonomies in the Developer Portal, which dictate the categorization of APIs and Products, and how they are displayed. The taxonomies that you define can be used by API developers to categorize APIs and Products in YAML files.
- **Implementing an image carousel**
  You can customize your Developer Portal home page to display images in a carousel. The image carousel provides your Developer Portal home page with a continuous slide show that is customized to your specifications.

- **Making your application's image public**
  You might want to display your application's image in an oauth authentication page to reassure customers that they are authenticating the correct application in the Developer Portal. That authentication uses a URL to the image that is stored on the Developer Portal. However, by default, those images are private unless you are logged in, which doesn't work if you want to display it in an oauth page. The solution is for the admin to change the **`application_image`** field to use public instead of private.
- **Providing navigation by tag hierarchy**
  You can provide the ability for users in the Developer Portal to navigate by using tag hierarchy.
- **Using the Views module in the Developer Portal**
  By using the Views module, you can fetch content from your Developer Portal site, and present it to users in different formats such as lists, graphs, and tables.

Configuring the front page

You can configure the Developer Portal front page.

Before you begin

You must have administrator access to complete this task.

About this task

Create a customized welcome page for your users. Note that blocks that are placed on the front page of the Developer Portal are visible to all users, regardless of whether the blocks have access restrictions placed on them. The visibility of blocks to all users also extends to the Featured Content block.

Procedure

To configure the front page, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure, then click Pages.
3. Alongside the welcome entry, click Edit.
4. Use the options provided to configure the front page.
   Here you can edit the page information, access, and configuration of the welcome page.
5. Click Update and save to save your changes.

What to do next

You can create a custom page. For more information, see Creating custom pages.

- **Disabling blogs**
  You can disable blogs on your Developer Portal.
- **Deleting forums from the front page**
  You can delete the help information on forums from the front page of your Developer Portal.

Related tasks

- Changing the front page banner block
- Changing the front page Featured Content block

Disabling blogs

You can disable blogs on your Developer Portal.

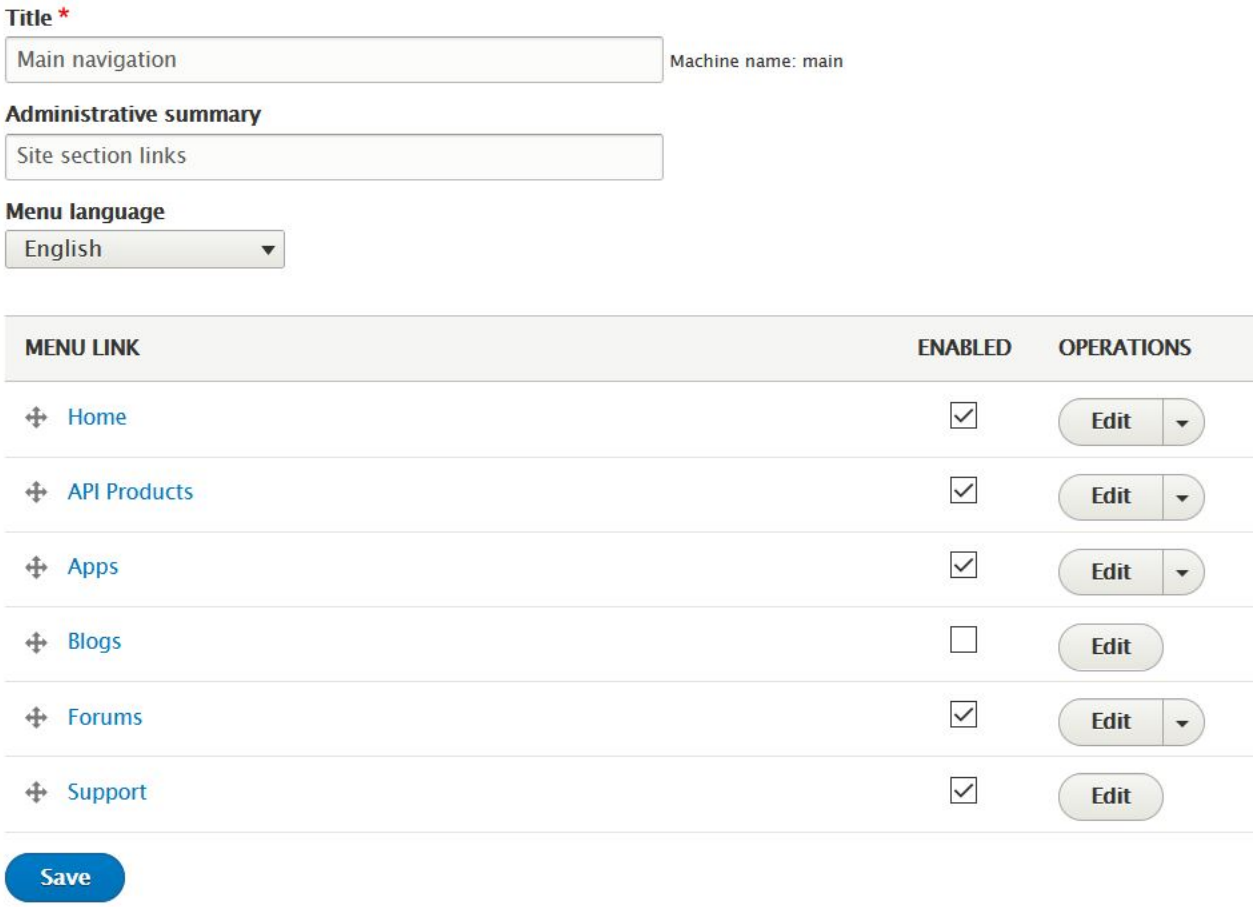
Before you begin

You must have administrator access to complete this task.

Before any changes are made, **`Blogs`** show as an option on the front page like this:

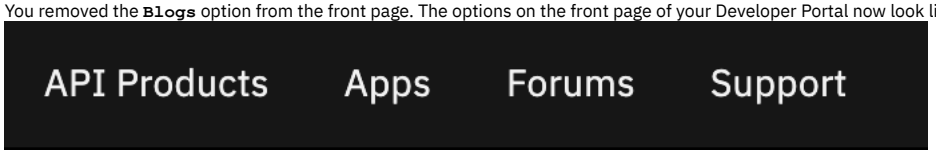
Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Menus from the administrator dashboard.
3. In the OPERATIONS column, click Edit menu for **`Main navigation`**.
4. Clear Enabled for the **`Blogs`** **`Menu link`**, then click Save.



Results

You removed the **`Blogs`** option from the front page. The options on the front page of your Developer Portal now look like this:



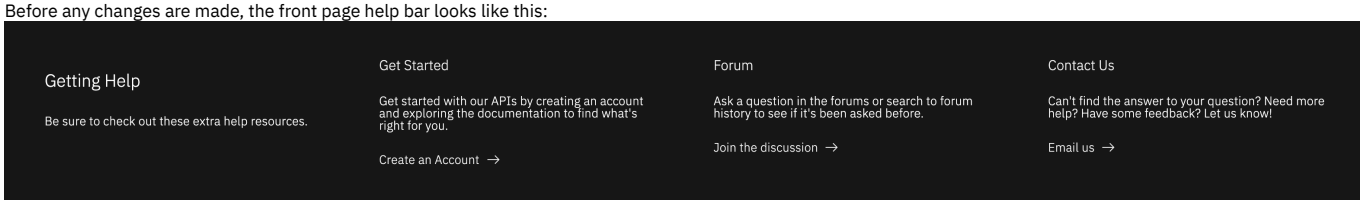
Deleting forums from the front page

You can delete the help information on forums from the front page of your Developer Portal.

Before you begin

You must have administrator access to complete this task.

Before any changes are made, the front page help bar looks like this:



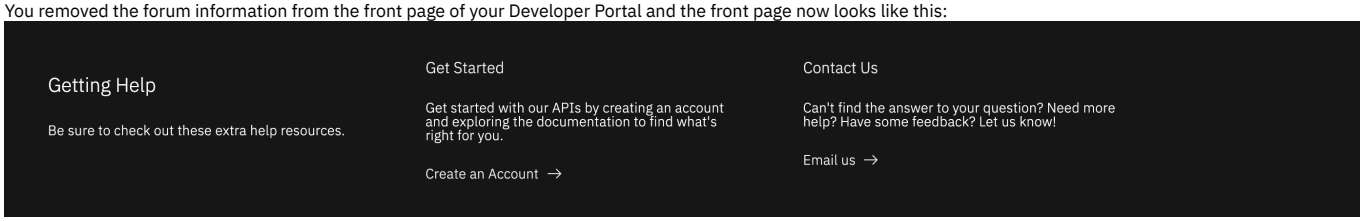
Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Block layout > Custom block library.
4. In Block description, enter `Get help`, then click Apply.

5. In the OPERATIONS column, click Edit for **`Get Help`**.
6. In the **`Body`** section, delete the forum information, then click Save.

Results

You removed the forum information from the front page of your Developer Portal and the front page now looks like this:



Adding a menu

You can add a menu to the Developer Portal, and define the items that are included in the menu.

Before you begin

You must have administrator access to complete this task.

Procedure

To add a menu, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Menus.
3. Click Add menu, provide a title, and an optional summary and menu language, and click Save.
4. To add items to your menu, click Add link, and configure the menu link details that you require.
5. Click Save to save your changes.
   A new block for your menu is created automatically, and your menu is now in the **`Menus`** list.

What to do next

You can use block configuration options to control which pages your menu is displayed on, and where it is positioned. You can use your menu with, or instead of, other menus available within the site. For more information, see Changing the blocks displayed on Developer Portal pages.

You can also update an existing menu, or change your menu. For more information, see Changing the items in the main menu.

Adding and changing the blocks displayed on Developer Portal pages

You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages.

Before you begin

You must have administrator access to complete this task.

About this task

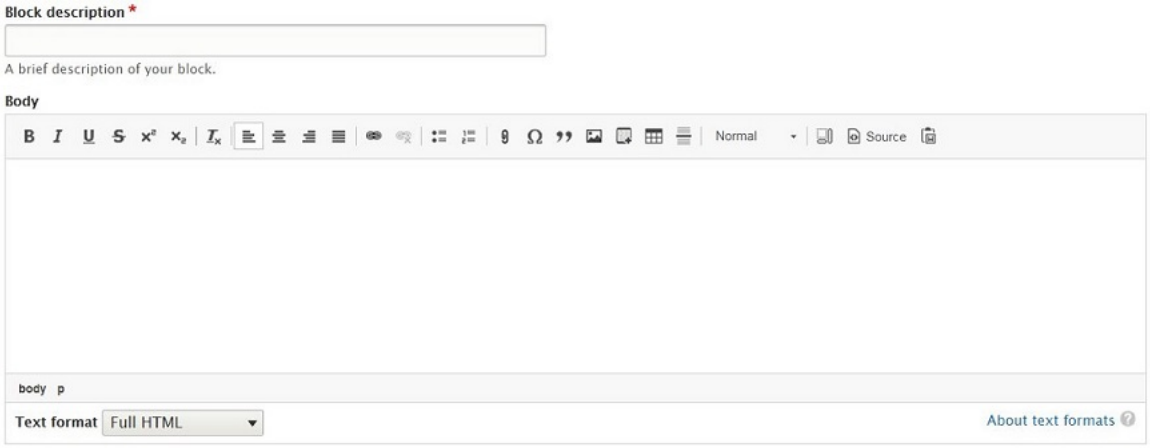
Blocks are boxes of content that are rendered into an area, or region, of a page. The content can be custom HTML, Github Markdown, or plain text, and you can specify the appearance, size, position, and also configure which pages a block appears on by editing its visibility settings. You can also modify an existing block to change its appearance, size, position, and visibility.

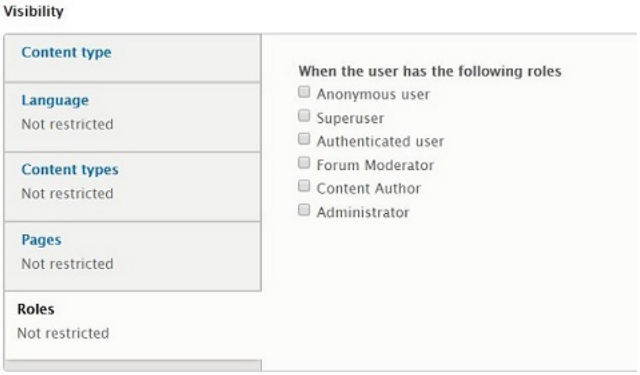
- Add a new block
- Edit the content of an existing block
- Modify the layout or visibility of an existing block

Procedure

To add a new basic block, complete the following steps.

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure, then click Block layout.
   All of the blocks that are in the system are displayed by theme. From the Block layout page you can assign blocks to a region, and control the order of the blocks within the region. Blocks are positioned on a per-theme basis, so if you enable more than one theme on your site, you can place blocks differently for each theme.
3. Click Place block > Add custom block.
   The Add custom block page is displayed.
4. Complete the Block description field.
5. Enter the content that you require for your block into the Body section.
   You can set the text format to Full HTML, Basic HTML, Restricted HTML, or Github Markdown. You can use the edit icons in the Body section to configure your content. Click the Source icon

   to enter code source, such as HTML.
   

6. Click Save.
   The new custom block is saved and the Configure block page is displayed.
7. Complete the block Title field.
8. In the Visibility section, specify how you want the block to be displayed.
   You can specify the content types, configure the language settings, restrict the block to show only on pages that display specific content types, set which pages the block is displayed on, and restrict which user roles the block is visible to.
   

9. In the Region section, select the region where the block should be displayed.
10. Click Save block.
    The new custom block is added to your Developer Portal site.

To edit the content of an existing block, from the Block layout page complete the following steps.

11. Click the Custom block library tab (or select Structure > Block layout > Custom block library from the administrator dashboard).
12. Click Edit alongside the required block.
13. Change the block description and body content as required.
14. Click Save to save your changes.

To modify the layout or visibility settings of an existing block, from the Block layout page complete the following steps.

15. To change the region in which a block is positioned, select the required Region from the drop-down list next to the required block.
16. To change the vertical sort-order of a block within a region, drag the block to the required position.
17. To edit the visibility settings of an existing block, click Configure from the Operations drop-down menu next to the required block.
18. To enable or disable an existing block, select Enable or Disable from the Operations drop-down menu next to the required block.
19. Click Save blocks to save your changes.

Changing the front page banner block

You can change the front page banner content and image that is displayed on the home page when a user first logs in to the Developer Portal.

Before you begin

You must have administrator or content author access to complete this task.

About this task

You can change the front page banner by editing the Welcome Banner block in the custom block library.

Procedure

To change the front page banner block, complete the following steps.

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure > Block layout.
3. Select the Custom block library tab.
4. Click edit against the Welcome Banner block entry.
   The Edit Basic block Welcome Banner page is displayed.
5. Edit the Body section with the content and image that you require.
   You can set the text format to Full HTML, Basic HTML, Restricted HTML, or Github Markdown. You can use the edit icons in the Body section to configure your content. Click the Source icon

   to view and edit the code source, such as HTML.
6. Configure the language and translation settings as required.
7. Click Save to save your changes.

Results

You successfully updated the front page banner block.

Related tasks

- Configuring the front page
- Adding and changing the blocks displayed on Developer Portal pages

Changing the front page Featured Content block

You can display featured Products or APIs on the front page of your Developer Portal by configuring the Featured Content block.

Before you begin

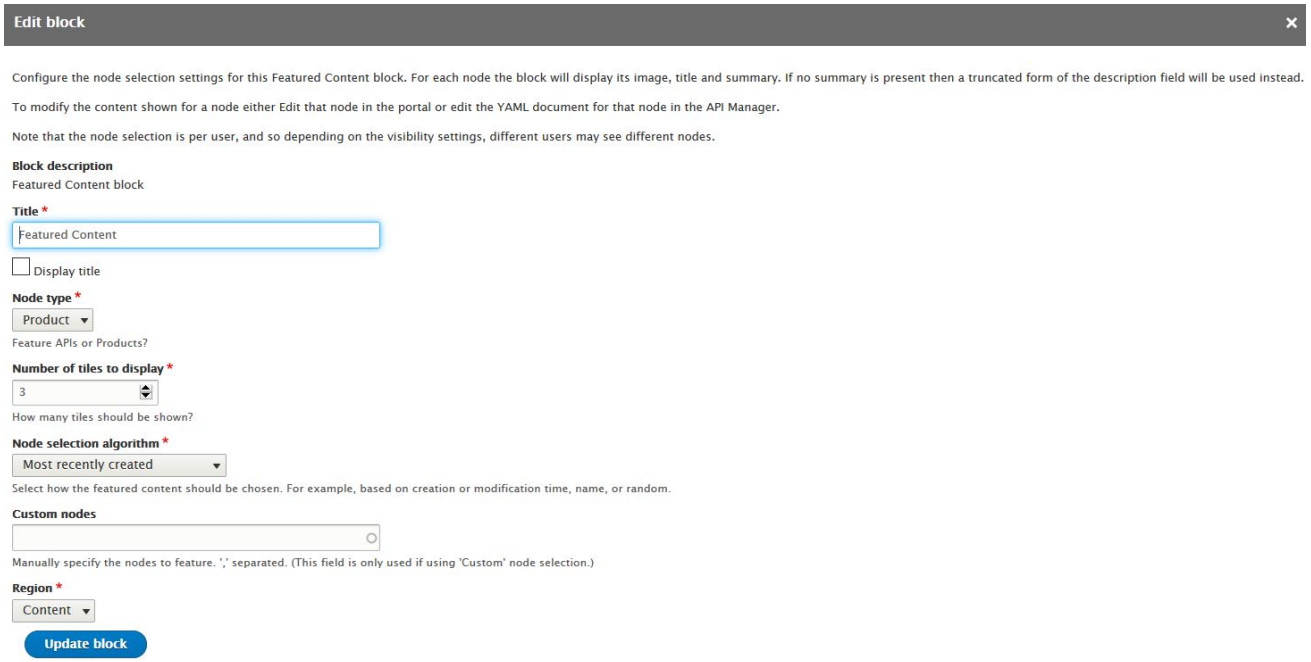
You must have administrator access to complete this task.

About this task

You can configure the Featured Content block on the front page of your Developer Portal. By default this block displays a list of the most recently created Products, but you can configure the block to display Products or APIs, and select how the featured content is chosen.

Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Pages.
4. Click Edit against the welcome page.
   The edit menu for the Welcome page is displayed.
5. Select Panels > Content from the navigation column, and then click Edit against the Featured Content block.
   The Edit block page for the Featured Content block is displayed.



6. Configure the Feature Content block as required.
   You can choose to display the title or not, select whether to use Products or APIs, for your node type. You can choose how many tiles to display, and select which algorithm is used to select the node. If you start typing in the **`Custom nodes`** field, you can choose from the options you are given.
7. Click Update block, then Update and save, to save your changes.

Results

You successfully configured the Featured Content block on the front page of your Developer Portal.

Changing the items in the main menu

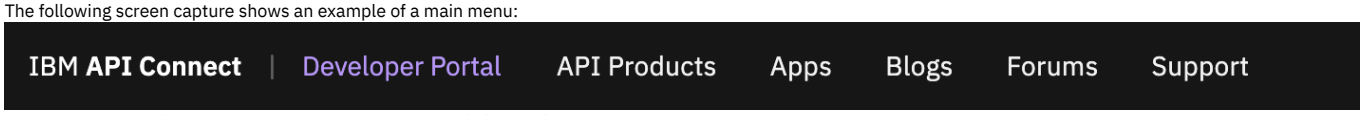
You can change the items in the main menu that is displayed on all pages in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

The following screen capture shows an example of a main menu:

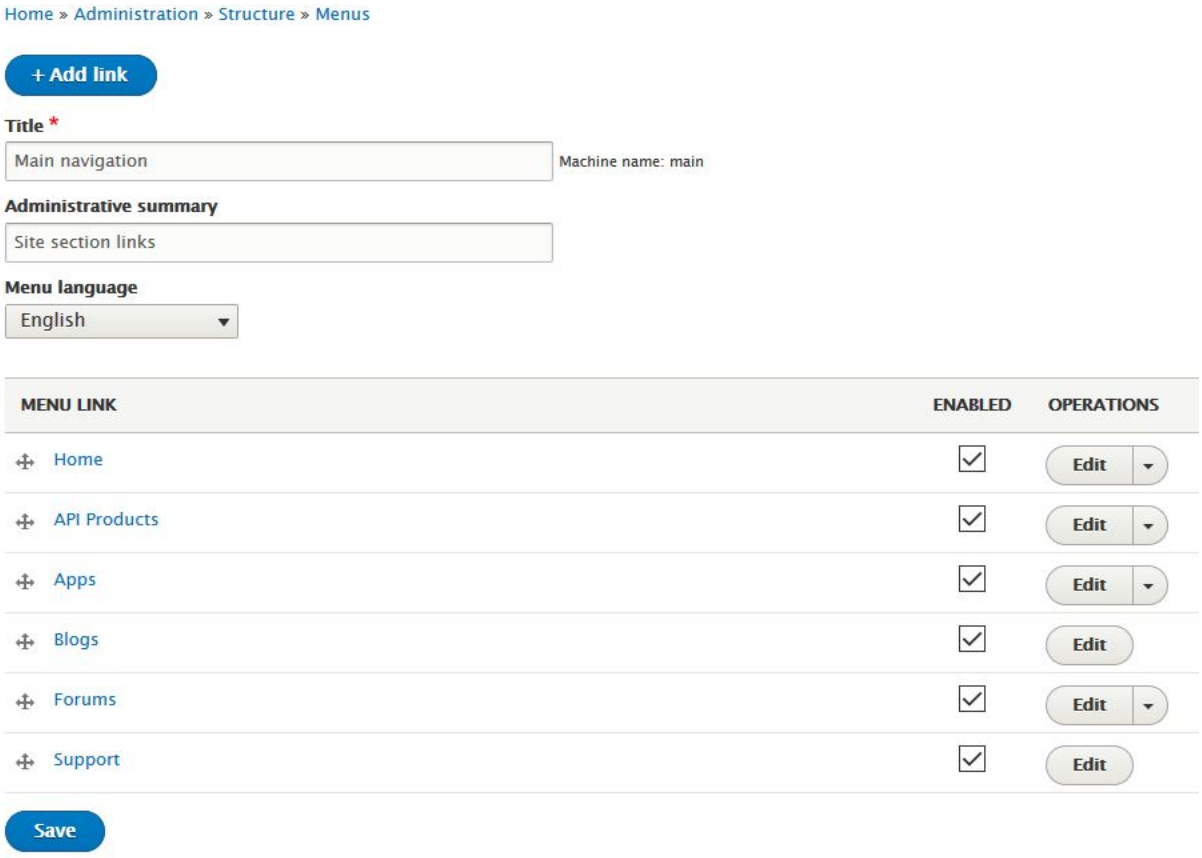


However, you can add new menu items, remove items, and change the item order.

Procedure

To change the items in the main menu, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Menus > Main navigation.



4. Use the options provided to change the menu items.
5. Click Save to save your changes.

Changing the ratings display

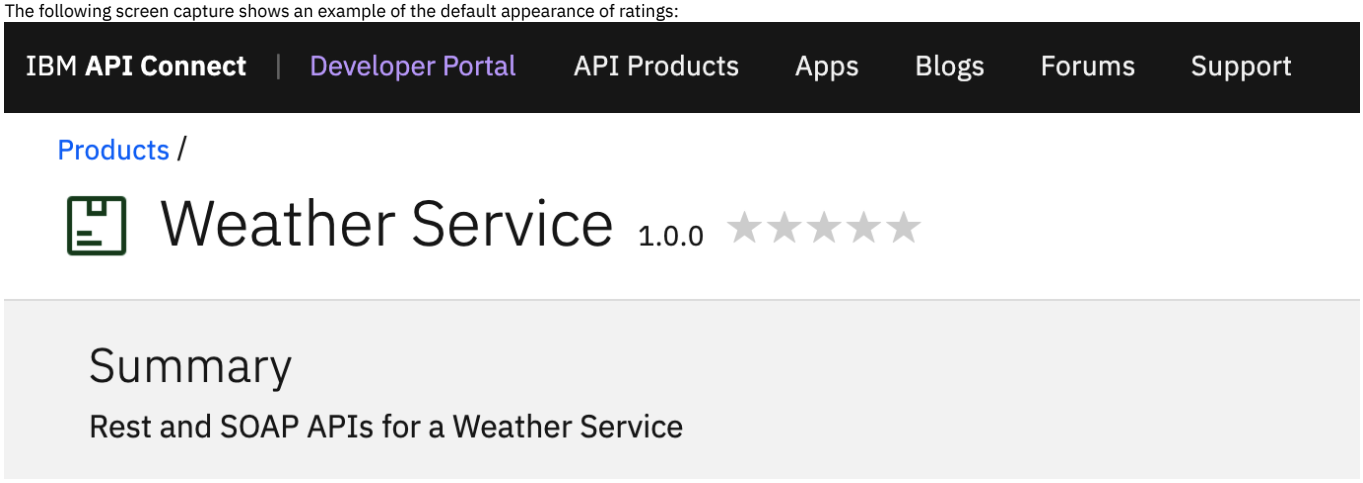
You can change the appearance of the content ratings in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

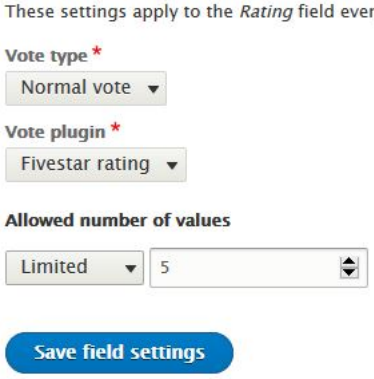
The following screen capture shows an example of the default appearance of ratings:



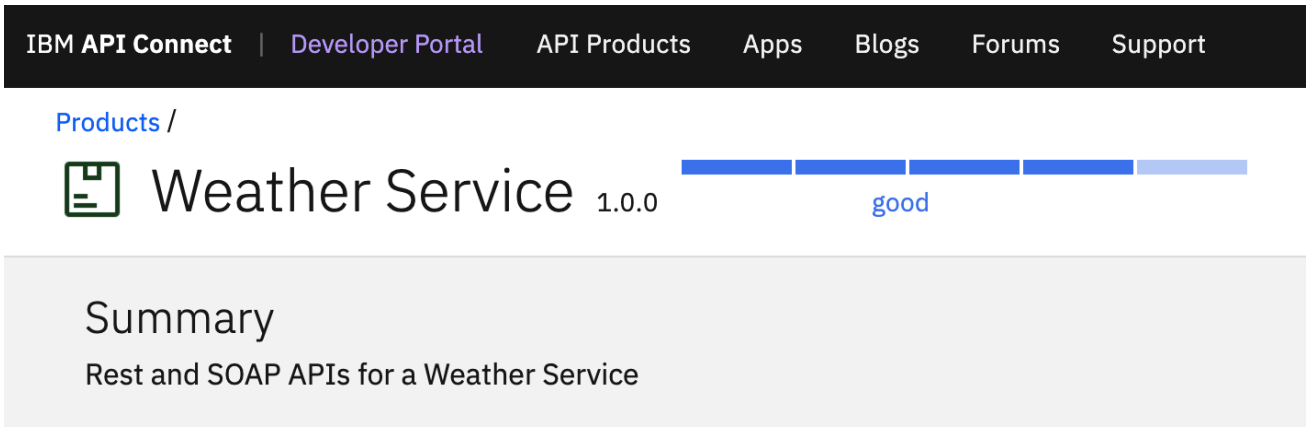
You can change this display if you want to.

Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Structure > Content types.
4. Click the Manage fields drop-down against the content type you want to change, for example **`Product`**.
5. Click Edit on the **`Rating`** row. Then, select the Field settings tab.



6. Change the style settings as required and click Save field settings.
   Here is an example of an alternative Ratings style:



Results

You successfully changed the ratings display for a content type in your Developer Portal.
Note: There are different view modes, and the settings would need to be changed for each view mode for each content type.

Displaying APIs and Products in categories

You can create taxonomies in the Developer Portal, which dictate the categorization of APIs and Products, and how they are displayed. The taxonomies that you define can be used by API developers to categorize APIs and Products in YAML files.

You might want to create taxonomies to control which set of APIs and products can and cannot be used in the Developer Portal.

By default, all taxonomies must be created in the Developer Portal. For more information, see Managing tags in the Developer Portal.

After you create your taxonomies for your APIs and Products, API developers can assign content to those taxonomies manually in the Developer Portal, or by using categories in YAML files.

However, if you create categories for your APIs and Products in the API Designer or API Manager UI, you can have your APIs and Products that are displayed in those categories in the Developer Portal. For more information about how to create categories for your APIs and Products in the API Designer or API Manager UI, see Organizing your Products into categories. For information about how to have your APIs and Products that are displayed in those categories in the Developer Portal, see Enabling dynamic category creation.

You can add more Developer Portal taxonomies to the categories that are defined in the API Designer or API Manager UI.

You can also provide the ability for users in the Developer Portal to navigate by using tag hierarchy. For more information, see Providing navigation by tag hierarchy.

- **Enabling dynamic category creation**
  You can display APIs and Products in pre-defined categories in the Developer Portal. You can define the categories that the APIs and Products are displayed in, in the API Designer or API Manager UI.

Enabling dynamic category creation

You can display APIs and Products in pre-defined categories in the Developer Portal. You can define the categories that the APIs and Products are displayed in, in the API Designer or API Manager UI.

Before you begin

You must have administrator access to complete this task.

Your APIs and Products must be categorized in the API Designer or API Manager UI, and they must be published. For more information, see Organizing your APIs into categories and Organizing your Products into categories.

About this task

By organizing your APIs and Products into categories, you can provide a hierarchical display for your APIs and Products in the Developer Portal.
Note: Organizing APIs and Products into a hierarchical view in the API Manager UI is different from tagging in the Developer Portal.
The categories that are defined in the API Designer or API Manager UI can be overridden by creating taxonomies in the Developer Portal. For more information, see Displaying APIs and Products in categories.

Procedure

1. To enable the Developer Portal to display the categories for your APIs and Products:
   a. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
   b. In the Categories section, select the Create taxonomies for categories if they do not already exist check box.
   c. Click Save configuration.
2. To enable the Developer Portal to display the change to the categories you made in step 1, you must republish your APIs and products.

Results

You enabled the Developer Portal to display the categories that you defined for your APIs and Products. You can see the categories that you defined, including the number of APIs and Products that are in each category, by selecting your APIs and Products in the Developer Portal.

Implementing an image carousel

You can customize your Developer Portal home page to display images in a carousel. The image carousel provides your Developer Portal home page with a continuous slide show that is customized to your specifications.

Before you begin

You must have administrator access to complete this task.

About this task

By implementing an image carousel, you can replace the default welcome banner or any previously set images.

Procedure

1. Create a content type of Image for your carousel.
   a. Click Structure > Content types > Add content type.
   b. Enter the name of your content type in the Name field.
      For example, *Slide show picture*.
   c. Optional: Specify the description of your content type in the Description field.
      For example, *A picture to display in the slideshow*.
   d. Click Save and manage fields for your newly created content type.
   e. Click Add field.
   f. Select Image for Add a new field, and enter a label for your field.
      For example, *slide image*.
   g. Click Save and continue.
      The Field settings page for your new field is displayed.
   h. In the DEFAULT IMAGE section, click Choose file to assign the image that you want as your default image for your slide show if no other images are found.
   i. Optional: Complete the Alternative text and Title fields.
   j. Click Save field settings.
   k. In the Manage fields tab for your newly created content type of *Slide show picture*, delete the `Body` field by clicking Delete on the OPERATIONS drop-down menu. Click Delete again to confirm.
2. Upload the images that you want to appear in your carousel.
   a. Click Content > Add content, then click the content type that you added in step 1.
   b. Enter a title for your image in the Title field.
      For example, if you had an image of a lighthouse, you can label the content as *Lighthouse*.
   c. Click Choose file under Slide image and select an image for your slide show. Complete the Alternative text field with some text to use when the image cannot be loaded.
   d. Click Save.
   e. Repeat sub-steps 2.a to 2.d until you are satisfied with the content of the slides for your carousel.
3. Create a view for your carousel.
   a. Click Structure > Views > Add new view.
   b. Complete View name.

For example, Slide show.
c. Select the Create a block check box in the BLOCK SETTINGS section.
d. Select Slick Carousel for the display format.
e. Enter the number of slides you have for your carousel in the Items per block field, then click Save and edit.
The Edit page for your slide show view is displayed.
f. Click the title of the view and enter `<none>` into the text field. Click Apply.
g. In the FIELDS section click Add, select the check box for slide image, and click Add and configure fields.
h. Ensure the check box for Create a label is not selected, and click Apply.
i. In the FIELDS section, click Content: Title > Remove.
j. In the FORMAT section, click Settings for Slick Carousel, and then set Skin Main to Default.
k. Scroll down to the CAPTION FIELDS section, and select the Content: slide image and Override main optionset check boxes.
l. In the OVERRIDABLE OPTIONS section, select the Autoplay, Dots, and Draggable check boxes.
m. Click Apply.
n. In the FILTER CRITERIA section, click Content: Published > Remove.
o. Click Save to save the Slide show view.
4. Configure your home page to host the carousel.
a. Click Structure, then click Pages.
b. Click edit for the welcome page, then click content in the Panels section.
c. To remove the current welcome banner, click the down-arrow under the OPERATIONS column for the Welcome Banner, and select Delete. Click Delete again to confirm.
d. To add your carousel, click Add new block and select the slide show view you created in step 3. Deselect Display title and click Add block.
e. Click Update and save.

Results

Your Developer Portal front page now has a slide show carousel.

Making your application's image public

You might want to display your application's image in an oauth authentication page to reassure customers that they are authenticating the correct application in the Developer Portal. That authentication uses a URL to the image that is stored on the Developer Portal. However, by default, those images are private unless you are logged in, which doesn't work if you want to display it in an oauth page. The solution is for the admin to change the **`application_image`** field to use public instead of private.

Before you begin

You must have administrator access to complete this task.

About this task

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Content types > Application > Manage fields
4. Navigate to the Image label, then in the **`Edit`** drop down, select **`storage settings`**.
5. Click **`Public files`** for **`Upload destination`**.
6. Click **`Save field settings`**.

Providing navigation by tag hierarchy

You can provide the ability for users in the Developer Portal to navigate by using tag hierarchy.

Before you begin

You must have administrator access to complete this task.

About this task

The Categories block provides the ability to navigate by tag hierarchy. The Categories block is disabled by default, so you need to enable the block and then configure it to your requirements.

Procedure

To provide navigation by tag hierarchy, complete the following steps:

1. Click Extend, in the filter box enter `Hierarchical Taxonomy Menu`.
2. Select **`Hierarchical Taxonomy Menu`**, then click Enable.
3. Click Structure > Block layout.
All of the blocks in the connect_theme are displayed.
4. Scroll to the region that you want to add the block to, for example Collapse right, click Place block.
5. Find the **`Hierarchical Taxonomy Menu`** block in the list and click Place block.
6. Complete any configuration tasks that you require for the **`Collapse right`** block, including selecting the correct taxonomy to use and indicating what pages to display the block on.

7. Click Save block to save your changes.
   You can add new blocks, and modify the existing blocks that are displayed on Developer Portal pages. For more information, see Adding and changing the blocks displayed on Developer Portal pages.

Related tasks

- Editing tags for a specific item
- Managing tags in the Developer Portal

Using the Views module in the Developer Portal

By using the Views module, you can fetch content from your Developer Portal site, and present it to users in different formats such as lists, graphs, and tables.

The Views module is a powerful SQL query builder that can access almost all the information in your Developer Portal site database and display it in any format.

There are many uses cases for views, but the following list shows some of the common ones:

- You want to display some of the content differently, such as Products, APIs, and applications.
- You want to display a block with the five most recent posts of some kind.
- You want to change the way that articles are displayed.
- You want to provide an unread forum posts list.
- You want to provide a monthly archive of posts.

For information about how to create a view in the Developer Portal, see the following topic. You can also follow a tutorial about creating a custom sort order view; see Tutorial: Using a custom weighting sort order on the product list page.

- **Creating views in the Developer Portal**
  You can create new views in the Developer Portal, such as content lists of Products, APIs, and applications, by using the Views module.
- **Configuring the search results view in the Developer Portal**
  You can configure the way that the search results are displayed in the Developer Portal.
- **Configuring the default number of items in a view list in the Developer Portal**
  You can configure the default number of items that are displayed in a view list in the Developer Portal.

Creating views in the Developer Portal

You can create new views in the Developer Portal, such as content lists of Products, APIs, and applications, by using the Views module.

Before you begin

You must have administrator access to complete this task.

About this task

Creating views in the Developer Portal enables you to control the presentation of specific content for users. For more information about views, see Using the Views module in the Developer Portal.

Procedure

To create a view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Structure > Views > Add new view.
   The Add view page is displayed.
4. Complete the View name and, optionally, the Description.
5. Complete the VIEW SETTINGS section with details of what you want your view to show, and how you want it to be sorted.
   For example, to create a view of APIs that are sorted by title, select Show `Content`, of type `API`, and sorted by `Title`.
6. Select whether to Create a page, Create a block, or both, for your view.
   A page is a full-screen page in the Developer Portal. Whereas a block would be a site block that you can then place on whatever pages you want to show a smaller view of the content, for example the 10 newest APIs.
7. Complete the options that you require for your view. For example, the display format, the number of items per page or block, whether to include a pager, whether to create a menu link, and whether to include an RSS feed.
8. Click Save and edit.
   Your new view opens in the Edit tab.
9. Refine your view further.
   Here you can further modify the display of your view or add new displays. More options include formatting of the view, what fields to include in a table or grid format, filtering and sorting criteria, add header or footer details, as well as advanced options around contextual filters, relationships, and behavior when there are no results to display. The Edit tab also includes a preview area, so you can check your view before publishing.
10. Click Save to make your changes permanent.
    When saved, your new view is visible in the Structure > Views pane.

Results

You created a new view in the Developer Portal.

Configuring the search results view in the Developer Portal

You can configure the way that the search results are displayed in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

In the Developer Portal, the search results are displayed as a view, and so you can control the presentation of this search content. For more information about views, see Using the Views module in the Developer Portal.

Procedure

To configure the search results view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Structure > Views.
   The list of views is displayed.
4. Find the Search content view in the list, and click Edit.
5. Edit the Search content view settings as required.
   Here you can modify the display of the view or add new displays. For example, you can configure the formatting of the view, the filtering and sorting criteria, add header or footer details, as well as more advanced options around contextual filters, relationships, and behavior when there are no results to display. There is also a preview area so you can check the view before publishing.
6. Click Save to make your changes permanent.

Results

You updated the view of the search results in the Developer Portal.

What to do next

You can configure the search index and server that are used to generate the search results. For more information, see Configuring search indexes and servers.

Related information

- Drupal Views module

Configuring the default number of items in a view list in the Developer Portal

You can configure the default number of items that are displayed in a view list in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

In the Developer Portal, you can manage how many items in a list are shown for each page on a view. For more information about views, see Using the Views module in the Developer Portal.

Procedure

To configure the search results view, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Structure > Views.
   The list of views is displayed.
4. In the list, find the view name that you want to update, for example, Products, then click Edit.
5. Click Pager.
6. Select Display a specified number of items, click Apply.

7. Set the Items per page as required, click Apply.
8. Click Save to make your changes permanent.

Results

You updated the default number of items in a view of the search results in the Developer Portal.

What to do next

The Views module is a powerful SQL query builder that can access almost all the information in your Developer Portal site database and display it in any format. For more information, see Using the Views module in the Developer Portal.

Configuration

As an administrator, you can control multiple aspects of the Developer Portal configuration, including managing security and general configuration tasks.

To learn how to configure specific aspects of your Developer Portal site, use the following section links.

- **Configuring content moderation**
  By configuring content moderation, you can configure a review and approval process for content types in the Developer Portal. You can specify which roles can perform which actions by assigning them the appropriate permissions.
- **General configuration tasks**
  You can perform general configuration tasks in the Developer Portal as an administrator.
- **Managing Developer Portal security**
  You can manage multiple security elements in the Developer Portal.

Configuring content moderation

By configuring content moderation, you can configure a review and approval process for content types in the Developer Portal. You can specify which roles can perform which actions by assigning them the appropriate permissions.

Before you begin

An example use case for configuring content moderation would be for a professional documentation writer to review an APIs documentation before it is published on the Developer Portal. They might, for example, notify the API Developer of typographical errors, suggest improvements to operation descriptions, or attach an image file. Edits might then be made before the documentation was republished to the API and then published to the Developer Portal.

You must have a Developer Portal enabled, and you must have administrator access to complete this task.

About this task

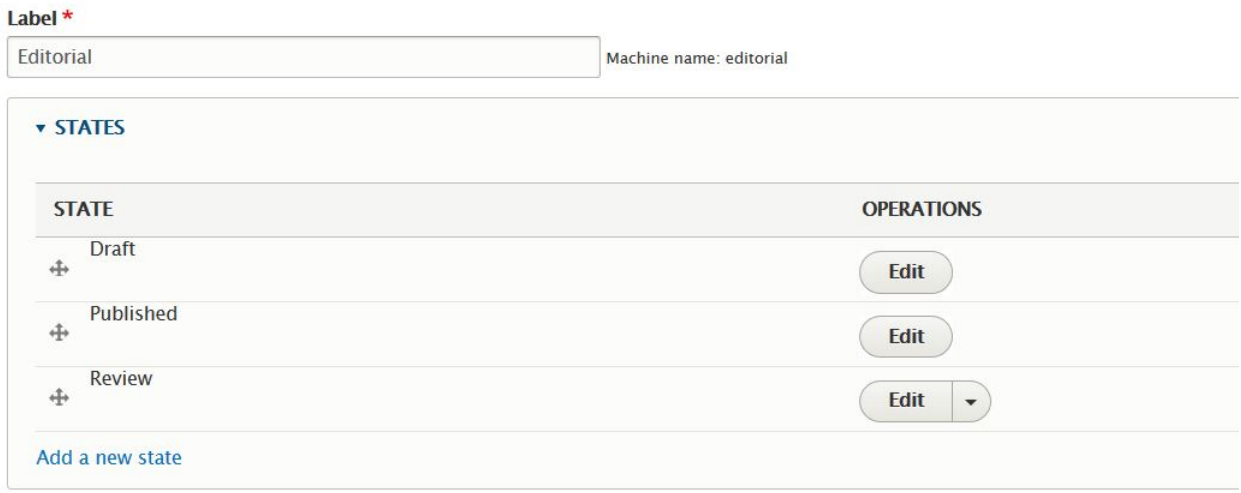
By using content moderation, you can configure the workflow of your Developer Portal site so that you can moderate content types, such as APIs and Products. However, you cannot moderate Consumer organizations or applications.

Procedure

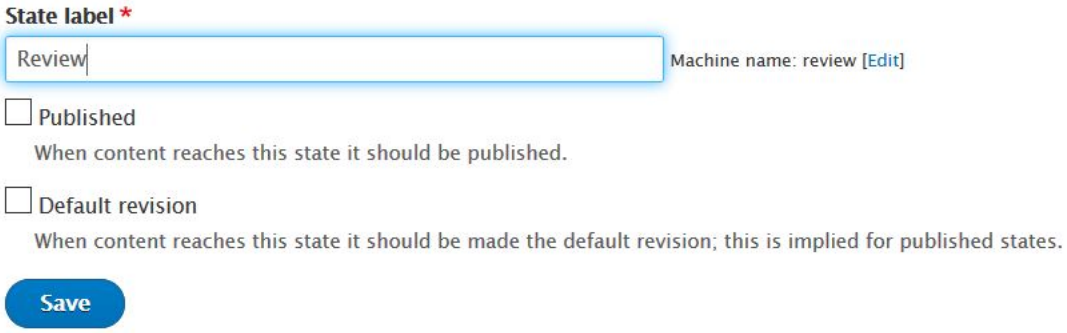
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Enable the required modules:
   a. On the administrator dashboard, click Extend.
   b. In the search bar enter **`workflows`**, then select the Workflows module and click Enable.
   c. In the search bar enter **`content moderation`**, then select the Content Moderation module and click Enable.
4. Navigate to Configuration > Workflow > Workflows
5. Check that **`Editorial`** is listed as a workflow. If it isn't listed:
   a. Click **`+ Add workflow`**.

   b. Enter **`Editorial`** as the label.
   c. Select the workflow type **`Content moderation`**.
   d. Click **`Save`**.
6. Set up the workflow:
   a. Click **`Edit`** on the **`Editorial`** workflow.

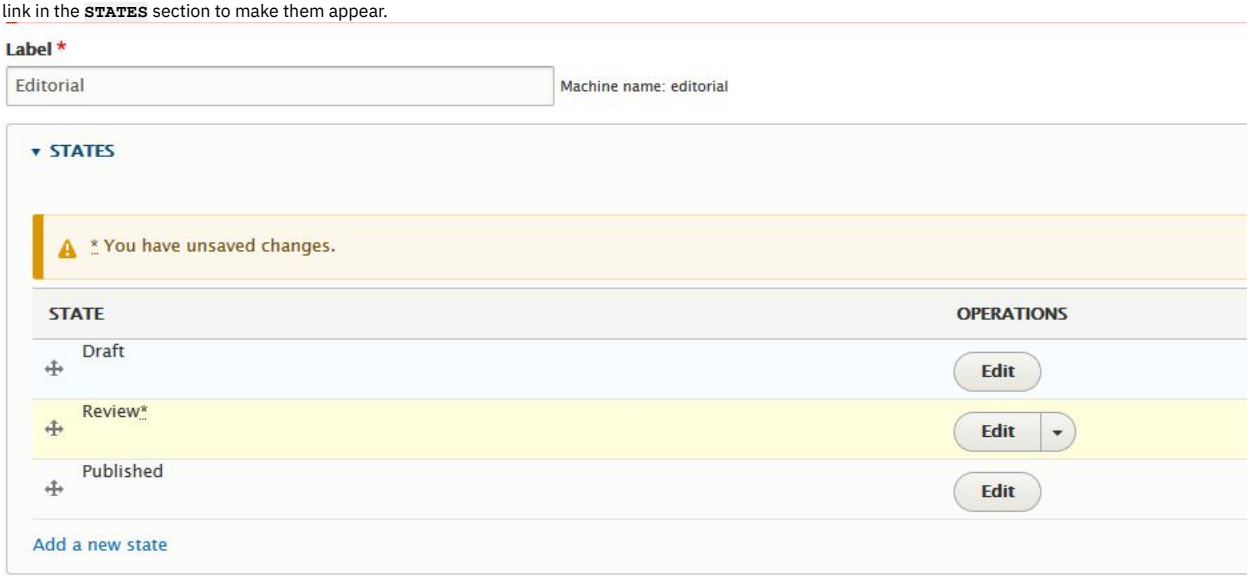
b. Click the **Add a new state** link.



c. Enter the value **Review** in the **State label** field, leave the **Published** and **Default revision** check boxes cleared then click Save.



d. Drag the **review** state to between the **Draft** and **Published** states then click 'Save'. If the drag handles are not visible, then click the **Hide row weights** link in the **STATES** section to make them appear.



e. Click the **Add a new transition** link.
f. Enter the value **Review** in the **Transition Label** field, tick the **Draft** check box in the **From** section and click **Review** in the **To** section then click Save.
g. Drag the **Review** transition to between the **Create New Draft** and **Publish** transitions, then click Save.
h. Click the **Edit** operation on the **Create New Draft** transition.
i. In the **From** check box list, untick **Published** and tick **Review** then click Save.
j. Click the **Edit** operation on the **Publish** transition.
k. In the **From** check box list, untick **Draft** and tick **Review** then click Save.
l. Click **Select** for the **Content types** item in the **THIS WORKFLOW APPLIES TO** section.
m. Select the **Product** content type then click Save.

7. Set up the permission schema for the workflow:
   a. Click People > Roles > Add a new role.
   b. In the **Role Name** field, enter the value **Editor** then click Save.
   c. Drag the **Editor** role to between the **Content Author** and **Administrator** roles then click Save.

d. Click the Permissions tab and scroll down to the **Content Moderation** permission section.
e. For the EDITOR role, select all the permissions in the **Content Moderation** section.
f. For the CONTENT AUTHOR role, select the permissions **Editorial workflow: Use Create New Draft transition**, **View any unpublished content**, and **View the latest version** then click Save permissions.
g. Ensure that the EDITOR and CONTENT AUTHOR roles have the permission **Use the administration toolbar**.

8. Assign the permissions to users in the portal. If you do not already have some registered users in your portal, create a new consumer organization and invite two new users to it.
   a. Click People.
   b. Tick the check box next to one of the users in the list, but not the admin user, then from the **Action** drop-down select **Add the Content Author role to the selected user(s)** then click **Apply to selected items**.
   c. Tick the check box next to another one of the users in the list, but not the admin user, then from the **Action** drop-down select **Add the Editor role to the selected user(s)** then click **Apply to selected items**.

Results

You successfully configured content moderation on your Developer Portal. You can now use the API manager user interface to publish a product to the portal, and transition the product from draft to review state.

- **Editing, reviewing, and publishing moderated content**
  If you have published content from API Manager UI to your Developer Portal or created content within the Developer Portal, you can edit and review the content through the Moderated content page before you publish it in the Developer Portal.

Editing, reviewing, and publishing moderated content

If you have published content from API Manager UI to your Developer Portal or created content within the Developer Portal, you can edit and review the content through the Moderated content page before you publish it in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

You must have content that is published to the Developer Portal from the API Manager UI.

Your role must have the necessary permissions that enable you to perform the tasks. For more information on assigning the necessary tasks, see Configuring content moderation.

About this task

In addition to editing content through the content moderation page, you can specify whether the content needs further review, or even publish the content if you have the permission.

Procedure

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. On the administrator dashboard, click Content.
4. Click the Moderated content tab.
5. If you have the necessary permissions to edit your content type, for example, you are a **Content author**:
   a. Click the Title of the content type that you want to edit, then click the Edit.
   b. Edit your content to your specification, then click Save.
      You are redirected to the **View** tab.
   c. When you have finished editing your content type on the **edit** tab, ensure that the **Review** state is selected from the **Change to** drop down, then click Save.
6. If you have the necessary permissions to review and publish your content type, for example, you are a **Reviewer**:
   a. On the administrator dashboard, click Content.
      The content page is displayed listing all content.
   b. Click the Moderated content tab.
   c. Click the Title of the content type that you want to review.
   d. Optional: If the content that you are reviewing requires editing, click Edit, apply the changes to your content type, then click Save.
      You are redirected to the View tab.

Results

You edited, reviewed, and published your draft content type.

General configuration tasks

You can perform general configuration tasks in the Developer Portal as an administrator.

To learn how to configure specific aspects of your Developer Portal site, use the following topic links.

- **Adding a page load progress indicator**
  You can extend your Developer Portal site by adding a page load progress indicator. This option might be useful on servers that have a slow connection to the API Manager, or for customers that are using OIDC and have slow response times from their OIDC provider.
- **Adding custom fields to user records**
  You can add custom fields to user records in the Developer Portal.
- **Checking the site status report**
  You can check the site status report in the Developer Portal.
- **Clearing the server caches**
  You can clear the server caches from within the Developer Portal.
- **Configuring a proxy**
  You can configure the proxy settings in your Developer Portal.
- **Configuring the visibility of analytics in the Developer Portal**
  How to configure the visibility of API consumer analytics in the Developer Portal.
- **Configuring Stripe in the Developer Portal**
  You can offer free use of your Product Plans to your API consumers, or you can offer Products with paid Plans. To enable API consumers to subscribe to Products with paid Plans, you must first configure the Stripe payment method in your Developer Portal.
- **Configuring cron to run scheduled tasks**
  You can configure cron to run scheduled tasks in the Developer Portal.
- **Configuring search indexes and servers**
  You can configure search indexes and servers in the Developer Portal by using the Search API module.
- **Configuring the date and time**
  You can configure the date and time in the Developer Portal.
- **Configuring the notifications event log settings**
  You can configure the settings of the notifications event log for the activity feed on your Developer Portal.
- **Configuring the settings of a Consumer organization**
  You can configure the settings of your Consumer organization from within your Developer Portal. The default settings allow users to delete their accounts and organizations, create more consumer organizations, rename their organization, and change the owner of their organization.
- **Configuring the site default timezone**
  You can configure the site default timezone in the Developer Portal.
- **Configuring the site error handling**
  You can configure the site error handling in the Developer Portal.
- **Configuring which buttons are displayed in the WYSIWYG rich text editor**
  You can configure which buttons are displayed in the WYSIWYG rich text editor in the Developer Portal.
- **Configuring which languages are available**
  You can configure which languages are available in the Developer Portal.
- **Customizing user account settings**
  You can customize the user account options that are displayed to users when they register and use the Developer Portal.
- **Disabling languages**
  You can extend your Developer Portal site by managing which languages are available for use on the site.
- **Disabling test tool restrictions**
  The default OAuth provider contains security mechanisms that prevent the ability to obtain authorization tokens in the Developer Portal test tool when you use Implicit or Authorization Code grant types. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option.
- **Enabling a cookie compliance banner**
  You can add a cookie compliance banner to your Developer Portal site by configuring the EU Cookie Compliance module.
- **Enabling code languages for code snippets**
  You can specify the languages that code snippets for APIs can be displayed in.
- **Enabling code snippets for SOAP APIs**
  By default, code snippets are only shown for REST APIs. You can enable code snippets for SOAP APIs.
- **Forcing new users to accept terms and conditions**
  You can extend your Developer Portal site by forcing new users to accept the terms and conditions before they are able to create accounts.
- **Hiding the admin registry on the login form**
  You can hide the admin registry as an option on the login form. The use case for this option might be a public facing Developer Portal where you want to remove the admin access from customer view.
- **Hiding the certificate in the header for APIs secured with mutual TLS**
  You can hide the certificate in the header for APIs secured with mutual TLS. By default, the `x-client-certificate` header is shown in an API when mutual TLS is configured. You can choose to turn off this option in your Developer Portal.
- **Importing and exporting taxonomies**
  You can import and export taxonomies in the Developer Portal.
- **Restricting access by IP address**
  You can restrict access to a role or a user by IP address in the Developer Portal.
- **Restricting or preventing access from search engines**
  You can restrict or prevent access to your Developer Portal from search engines. You might want to control the access if you have a development or test site.
- **Toggling the site in and out of maintenance mode**
  You can put the Developer Portal site into maintenance mode for short periods of time. During maintenance mode, only an administrator is able to access the site, and all other users who enter the site URL get a maintenance message set by the administrator.
- **Viewing available updates**
  You can view available updates in the Developer Portal.

Adding a page load progress indicator

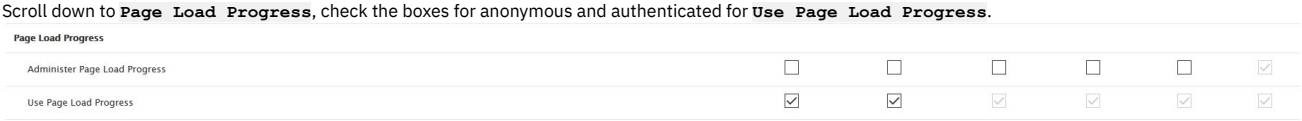
You can extend your Developer Portal site by adding a page load progress indicator. This option might be useful on servers that have a slow connection to the API Manager, or for customers that are using OIDC and have slow response times from their OIDC provider.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected check box. The disabled modules are displayed without a selected check box.
3. Enter `page_load_progress` into the search filter.
   The **`Page Load Progress`** content item is displayed in the list of content.
4. By default, the **`Page Load Progress`** module is enabled. However, if not, select the check box and click Enable.
   You have now enabled the default settings for the **`Page Load Progress`** module.
5. To change the default settings for the module, navigate to Configuration > User interface > Page Load Progress.
6. Decide which settings you want to change, for example, **`Time to wait before locking the screen`**, and click Save.
   Note: Changing the settings does not apply to the OIDC buttons. If it is enabled, the OIDC buttons use the page load process indicator, but do not allow any configuration.
7. To enable the permissions, navigate to People > Permissions.
8. Scroll down to **`Page Load Progress`**, check the boxes for anonymous and authenticated for **`Use Page Load Progress`**.



9. Click Save permissions.

Results

You successfully enabled the **`Page Load Progress`**. Now, the page loading spinner is automatically added to the login and register forms for OIDC buttons.

What to do next

You can customize the appearance of the modal page and spinner in CSS in a custom theme.

Adding custom fields to user records

You can add custom fields to user records in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

Any data that is added to a custom field for user records will remain in the Developer Portal database.

Procedure

1. In the administrator dashboard, click Configuration > People > Account settings.
2. Click the MANAGE FIELDS tab.
3. Click Add field.
4. Specify the type of data that your field can store by selecting the type from the Add a new field list.
5. Enter a label for your field in the Label field.
6. Optional: If you want to edit the machine name that is automatically created based on your label, click Edit and enter the new name.
7. Click Save and continue.
8. Configure any additional values for the new field then click Save field settings.
   You have added a custom field to your user record.

What to do next

You can configure the contents of your custom field depending on what type of data you specified it could store.

Checking the site status report

You can check the site status report in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To check the site status report, complete the following steps:

1. Click Reports in the administrator dashboard.
2. Click Status report.
3. You can review information relating to the site status report from this view.

Clearing the server caches

You can clear the server caches from within the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

1. In the administrator dashboard, click Configuration, then in the Development section, click Performance.
2. Click Clear all caches.
   You have cleared the server caches, and a message will be displayed stating: `Caches cleared`.

Configuring a proxy

You can configure the proxy settings in your Developer Portal.

Before you begin

You must have administrator access to complete this task.

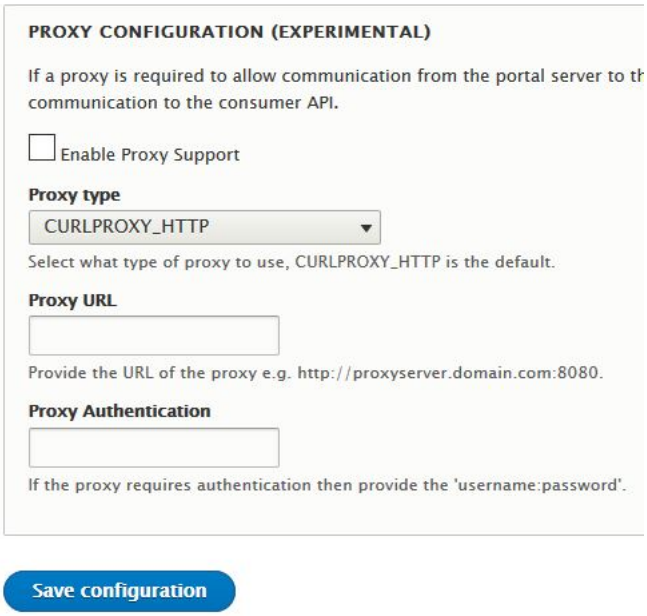
About this task

The Developer Portal communicates with the management server by way of the consumer and product APIs. For a Developer Portal that is deployed externally to the management server zone, and does not have access to the consumer and product APIs, a proxy can be used.

Procedure

To configure a proxy to use for communicating with management server APIs, complete the following steps:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > System > IBM API Connect in the administrator dashboard.
4. To configure a proxy, select **`Enable Proxy Support`**, and make the required changes.



5. When your changes are complete, click Save configuration.

Results

You successfully configured a proxy on your Developer Portal.

Configuring the visibility of analytics in the Developer Portal

How to configure the visibility of API consumer analytics in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Restriction:

- If the Catalog that is associated with this Developer Portal has two or more gateway services enabled then, for analytics data to be available, the gateway services must all be associated with the **same** analytics service. If the set of associated analytics services for the gateway services includes two or more different analytics services then the Developer Portal does **not** display analytics data. An analytics service is associated with a gateway service in the Cloud Manager user interface; see Associating an analytics service with a gateway service.
- If the visualizations in your dashboard views display no data, show data only for a time period in the past, or the Dashboard tab does not show up, a possible reason might be that access to analytics data within API Connect has been switched off. Contact your API provider for confirmation.

About this task

What API consumer analytics are displayed in the Developer Portal depends on the visibility configuration settings.

Procedure

- To configure the visibility of the API consumer analytics in Developer Portal, complete the following steps:
  1. If the administrator dashboard is not displayed, click Manage to display it.
  2. Click Configuration in the administrator dashboard.
  3. Under the System heading, click IBM API Developer Portal.
  4. Under the Configuration section, clear the Display Analytics checkbox to remove all the analytics links from the portal site. By default, the Display Analytics checkbox is selected and the analytics section is visible in the portal site.
  5. Click Save configuration to save the changes.
     You have configured the visibility of analytics in the Developer Portal.
- To configure the visibility of the API analytics charts, complete the following steps:
  1. If the administrator dashboard is not displayed, click Manage to display it.
  2. Click Configuration in the administrator dashboard.
  3. Under the System heading, click IBM API Developer Portal Consumer Analytics.
  4. From the list of Metrics and charts, clear the checkboxes corresponding to the charts that you want to remove from the analytics section in the portal site. By default, all the charts that are displayed under the metrics and charts section are visible in the analytics section.
  5. Click Save configuration to save the changes.
     You have configured the visibility of Metrics and charts in the analytics section.

Configuring Stripe in the Developer Portal

You can offer free use of your Product Plans to your API consumers, or you can offer Products with paid Plans. To enable API consumers to subscribe to Products with paid Plans, you must first configure the Stripe payment method in your Developer Portal.

Before you begin

You must have administrator access to complete this task.

To use Stripe as your credit card processing vendor, you must have port 443 open to HTTPS communication between the Stripe API and your Developer Portal management and the Management cluster servers. See Firewall requirements on Kubernetes and Firewall requirements on VMware for more information about this requirement.

The billing microservice must be enabled on your management system by your system administrator. See Configuring monetization on VMware and Configuring monetization on Kubernetes for more information.

Billing integration support must be configured in the API Manager. For more information, see Monetizing your Products.

About this task

API Connect includes a subscription billing feature that allows API providers to define pricing Plans in their API Products, and monetize their API offerings. If a Product contains a pricing Plan, API consumers must enter their payment information into the Developer Portal before they can subscribe to that Plan.

API Connect supports integration with Stripe Subscription Billing, an independent cloud service that manages monetized product Plans, customers, their payment information, and their subscription history, in order to generate monthly invoices and charge customers automatically. With this integration, Stripe serves as both the subscription billing system and the payment processing system.

Support is provided for Stripe in the Developer Portal, but before API consumers can enter their payment information, you need to configure the Stripe billing support in every Developer Portal where you will offer paid Plans. After this support is configured, API consumers can provide their payment information, and then subscribe to the paid Plans. Complete the following instructions to enable the APIC Monetization Stripe Integration module, and then configure your Stripe API credentials.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
3. Enter `stripe` into the search filter.
   The **`ibm_stripe_payment_method`** module called APIC Monetization Stripe Integration is displayed in the list of contents.
4. Select APIC Monetization Stripe Integration and click Enable.
   The default settings for the **`ibm_stripe_payment_method`** module are now enabled.
5. In the administrator dashboard, click Configuration > System > IBM API Developer Portal Billing.
   The IBM API Developer Portal Billing Settings page is displayed.
6. Change the drop-down option to be the newly enabled APIC Monetization Stripe Integration (ibm_stripe_payment_method) as the module to use to create payment methods for the Stripe billing provider. Click Save configuration.
7. In the administrator dashboard, click Configuration > System > IBM APIC Stripe Integration, and enter your Stripe API credentials. These credentials must match the Stripe API credentials for the billing integration resource that is attached to the Catalog for this Developer Portal.
   Refer to your Stripe dashboard to get your publishable key and secret key; see https://dashboard.stripe.com/apikeys.
   Note: Each Stripe account comes with two sets of API keys, one for testing, and one for production. Each set of API keys has a distinct namespace for Stripe objects. Test API keys cannot see objects created by production API keys, and vice versa. You cannot switch the API keys of one of your billing integrations with the keys from another account, or swap your test and production keys over, as that would prevent API Connect from resolving the Stripe objects that were created by using the old keys. If you want to use your Stripe test keys, you must create a separate Catalog for testing, and not add your Stripe test keys to production Catalogs.
8. Click Save integration.

Results

You successfully configured the Stripe payment method in the Developer Portal. API consumers can now add a payment method to their consumer organization, so that they can subscribe to Products with paid Plans.

What to do next

After both the Catalog and the Developer Portal are configured, you can create Products with pricing Plans. See Defining a Product with billing integration for more information.

Related information

- Monetizing your Products
- Considerations when changing a Product lifecycle with billing

Configuring cron to run scheduled tasks

You can configure cron to run scheduled tasks in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure cron to run scheduled tasks, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the SYSTEM heading, click Cron.
3. Use the drop-down list for the module you want to configure and select Edit.
4. From the Run cron every drop-down menu, select the frequency of cron runs for this module.
5. Click Save.
   You have configured cron to run scheduled tasks.

Configuring search indexes and servers

You can configure search indexes and servers in the Developer Portal by using the Search API module.

Before you begin

You must have administrator access to complete this task.

About this task

The Search API module provides a generic framework for search capabilities within the Developer Portal, and you can customize the default database server and content index setup that is provided. By default, the search in the Developer Portal runs over the entire OpenAPI and WSDL documentation. It is not possible to restrict the search to specific sections of an Open API or WSDL document. The search can be configured to include any custom fields that were added to the Developer Portal, but it does not include any attached files.
The search results page is a view in the Developer Portal, and so it can also be configured. For more information, see Configuring the search results view in the Developer Portal.

Procedure

To configure the default database server and content index settings, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > Search and metadata > Search API in the administrator dashboard.
4. To configure the server settings, click Edit against the Database Server and make the required changes. For example:
   - Server name; the displayed name for the server.
   - Description; a description of the server.
   - CONFIGURE DATABASE BACKEND; the minimum number of characters a word must consist of to be indexed, and whether to search on parts of a word. However, wildcard searching can make searches on large sites very slow.
   - AUTOCOMPLETE SETTINGS; how suggestions are computed when autocompletion is used.
5. To configure the index settings, click Edit against the Default content index and make the required changes. For example:
   - Index name; the displayed name for the index.
   - Data sources; the data sources of the items to be stored in the index.
   - CONFIGURE THE DATASOURCE; for each data source, set which bundles of content should be indexed, and in which languages.
   - Server; the server that the index should use.
   - Description; a description for the index.
   - INDEX OPTIONS; whether the index is read only, whether to index new or updated items immediately, and setting how many items to index when indexing items during a cron run.
6. When your changes are complete, click Save.
   The View page of the updated default database server or default content index is displayed.

Results

You successfully configured the search indexes and servers in the Developer Portal.

What to do next

You can configure how searches are displayed by editing the Search content view. For more information, see Configuring the search results view in the Developer Portal.

Related information

- Search API module

Configuring the date and time

You can configure the date and time in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure the date and time, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. Under the Region and language heading, click Date and time formats.
4. Click Edit for the format you want to change.
5. Change the settings of the format then click Save format.
   You have configured the date and time.

Configuring the notifications event log settings

You can configure the settings of the notifications event log for the activity feed on your Developer Portal.

About this task

The Developer Portal includes an activity feed that shows Consumer organization users the details of events that occur on the organization, such as application creation, subscriptions, and so on. This activity feed is displayed by default on the navigation bar of the organization, and as an additional tab on the organization page and the application page.

The default number of days that events are kept for is 30. You can edit this setting in the IBM APIC Notifications Settings page of your Developer Portal, as described in the following instructions.

Procedure

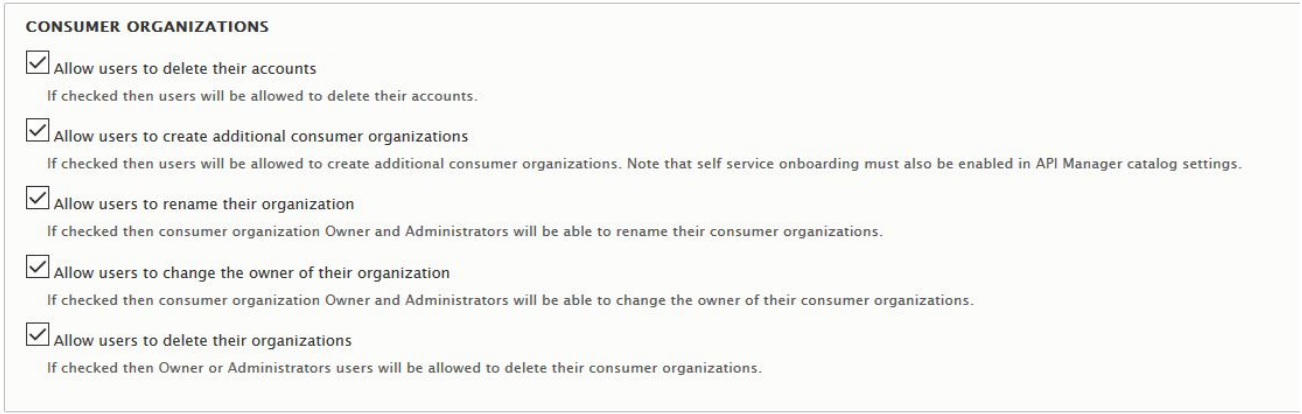
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > System > IBM APIC Notifications Settings.
   The IBM APIC Activity Feed Settings page is displayed.
4. Enter the required number of days to keep events for in the Retention Days field.
5. Click Save configuration.

Configuring the settings of a Consumer organization

You can configure the settings of your Consumer organization from within your Developer Portal. The default settings allow users to delete their accounts and organizations, create more consumer organizations, rename their organization, and change the owner of their organization.

Procedure

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > System > IBM API Connect.
4. Scroll down to **CONSUMER ORGANIZATIONS**.



5. Clear which ever options you want to change, click Save configuration.

Configuring the site default timezone

You can configure the site default timezone in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure which languages are available, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the REGIONAL AND LANGUAGE heading, click Regional settings.
3. From the Default Time Zone drop-down menu, select the default time zone.
4. Click Save configuration.
   You have configured the default time zone.

Configuring the site error handling

You can configure the site error handling in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure the site error handling, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the DEVELOPMENT heading, click Logging and errors.
3. You can review and configure all information relating to site error handling from this view.
   Note: On a production system, set Logging and errors to **`none`**, to avoid any security issues.
4. Click Save configuration.
   You have configured the site error handling.

Configuring which buttons are displayed in the WYSIWYG rich text editor

You can configure which buttons are displayed in the WYSIWYG rich text editor in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure which buttons are displayed in the WYSIWYG rich text editor, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration > Content authoring > Text formats and editors.
3. Click Configure for the Full HTML profile.
4. In the **`TOOLBAR CONFIGURATION`** section, edit the buttons that you want to expose.
5. Click Save configuration.
   You have configured the editor buttons that are displayed in the WYSIWYG rich text editor.

Configuring which languages are available

You can configure which languages are available in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure which languages are available, complete the following steps:

1. Click Configuration in the administrator dashboard.
2. Under the REGIONAL AND LANGUAGE heading, click Languages.
3. Under the DEFAULT heading, select the radio button for the language you want to set as default.
4. To add a new language, click +Add language. Use the drop-down menu to select the language you want to add.
5. Click Save configuration.
   You have configured the languages that are available.
6. You can also provide your own translations, for strings of the website that are not translated, from the User interface translation view, under Configuration > Regional and Language in the administrator dashboard.

Results

You have configured the available languages for the Developer Portal.
Note: Multilingual API and Product documentation can be created by using an **`x-ibm-languages`** extension directly in the OpenAPI definition. For more information, see Using **`x-ibm-languages`** to create multilingual API and Product documentation.

Customizing user account settings

You can customize the user account options that are displayed to users when they register and use the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can change what fields users of the Developer Portal see when they register to use the site, when they log in, and when they view their user account profile. For instance, you can change the field order display, hide fields, and add custom fields.
As an example, the Consumer organization field is a required entry field when a user creates an account on the Developer Portal. However, you can configure this field to be hidden, in which case the field is automatically completed with a default entry of **`firstname lastname`** of the new user instead.

Procedure

To customize the user account settings, complete the following steps.

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > People > Account settings in the administrator dashboard.
   The Account settings page is displayed.
4. From the Settings tab you can edit the following options:
   a. CONTACT SETTINGS; enable or disable the personal contact form for new users.
   b. PASSWORD RESET TIMEOUT; set the timeout in seconds for one-time login links. This setting applies only to the admin user of the Developer Portal.
   c. ANONYMOUS USERS; set the name to be used to indicate anonymous users.
   d. ADIMINISTRATOR ROLE; assign the administrator role that is automatically given new permissions whenever a new module is enabled. The default is Superuser.
      Note: The Superuser role has the highest level of permissions in the Developer Portal. This role bypasses all content access control, which means that users that have a Superuser role can see all of the site content. For more information, see Working with roles in the Developer Portal.
   e. LANGUAGE SETTINGS; enable or disable translation.
   f. REGISTRATION AND CANCELLATION; enable or disable the password strength indicator, and set the action to take when a user account is canceled.
   g. Notification email address; set the email address to use as the from address for all account notifications.
   h. Emails; edit the emails that are sent when an account is blocked or canceled, or when a new password is requested.
   i. Click Save configuration to save your updates.
5. Click the Manage fields tab to customize the fields that are available for storing user data.
   Here you can edit the settings for current fields, delete fields, and add new fields. Remember to save any updates that you make.
6. Click the Manage form display tab to configure how the user forms are displayed.
   a. Click the Default form mode to configure how the form fields display when a user profile is being edited.
   b. Click the Register form mode to configure how the form fields display when a new user creates an account.
      For each user form mode you can edit the field settings, enable and disable fields, and drag the fields to change their display order. You can also manage the form modes.
      For example, to hide the Consumer organization field when a new user creates an account, click the Register form mode, and drag the Consumer organization into the Disabled section.

   c. Click Save to save your updates.
7. Click the Manage display tab to configure how fields are displayed on a user profile page.
   You can edit the field settings, enable and disable fields, and drag the fields to change their display order. You can also manage the display view modes. Remember to save any updates that you make.
8. Click the Translate account settings tab to configure the language options for the user account settings. Remember to save any updates that you make.

Results

You successfully customized the user account settings in the Developer Portal.

Related tasks

- Configuring CAPTCHA

Disabling languages

You can extend your Developer Portal site by managing which languages are available for use on the site.

Before you begin

You must have administrator access to complete this task.

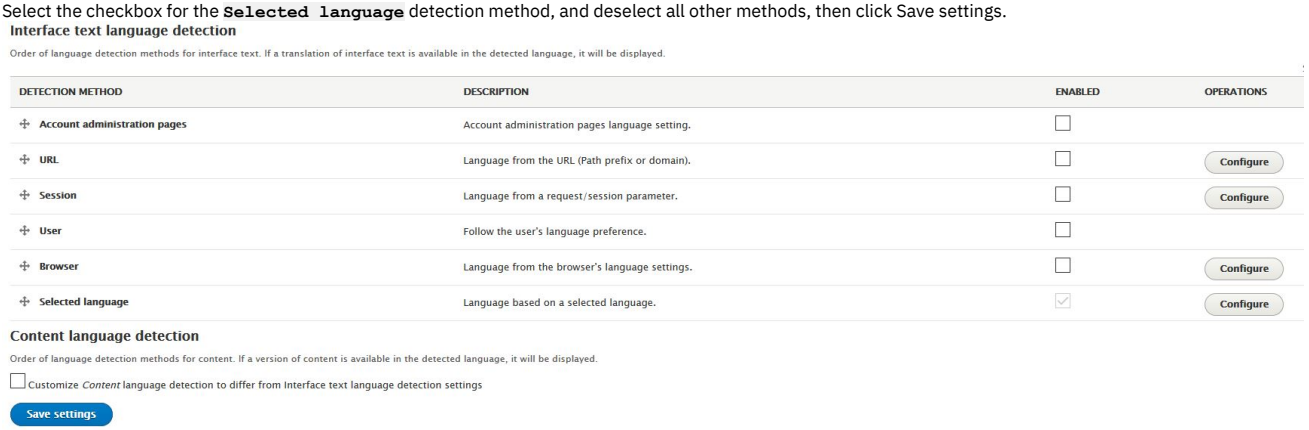
Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.

To disable all languages apart from English.

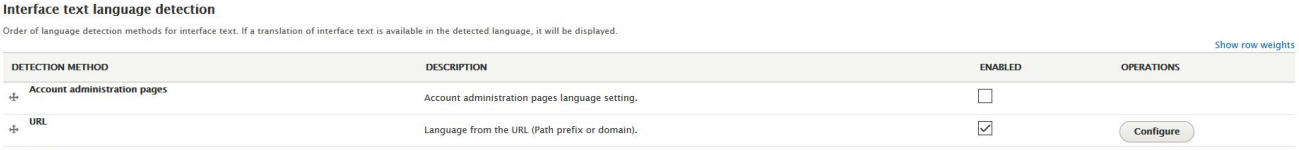
2. Navigate to Configuration > Regional and language > Languages > Detection and selection.

3. Select the checkbox for the **`Selected language`** detection method, and deselect all other methods, then click Save settings.

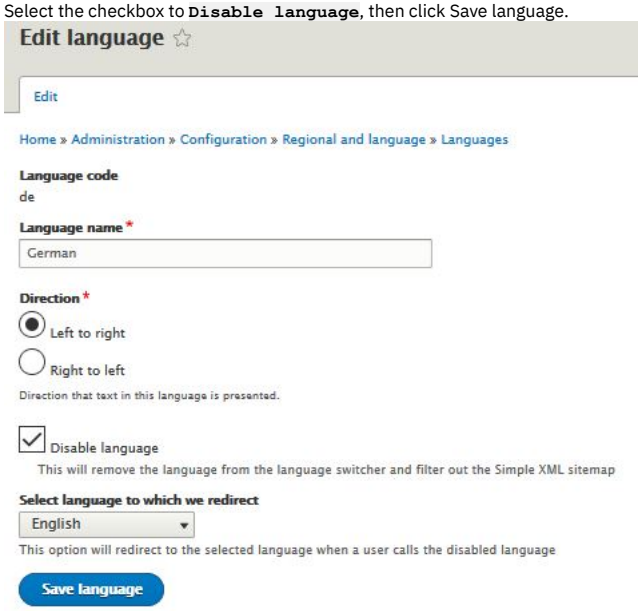


To disable one or more languages.

4. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
5. Enter `disable language` into the search filter.
   The Disable Language content item is displayed in the list of content.
6. Select the checkbox for the **`Disable Language`** module, and click Enable.
7. Navigate to Configuration > Regional and language > Languages > Detection and selection.
8. Select the checkbox to enable URL, and click Save settings.



9. Navigate to Configuration > Regional and language > Languages.
10. Click Edit on the language that you want to disable.
11. Select the checkbox to **`Disable language`**, then click Save language.



Results

You successfully disabled your chosen language.

Disabling test tool restrictions

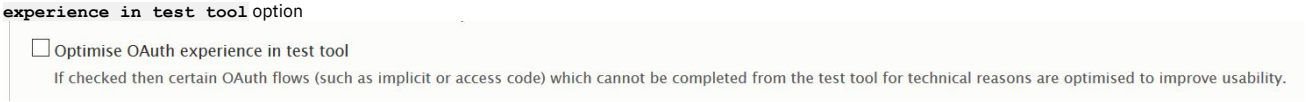
The default OAuth provider contains security mechanisms that prevent the ability to obtain authorization tokens in the Developer Portal test tool when you use Implicit or Authorization Code grant types. If your OAuth provider allows those actions to function correctly from the test tool, you can disable this option.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Configuration > System > IBM API Connect.
3. In the **CONFIGURATION** section, deselect the **Optimise OAuth experience in test tool** option

4. Click Save configuration.

Results

You successfully disabled the **Optimise OAuth experience in test tool** option. You can now use the Developer Portal test tool to test an Implicit or Authorization Code grant type in an OAuth provider API, if your OAuth provider allows those actions to function correctly from the test tool.

What to do next

You can enable the **Optimise OAuth experience in test tool** option by repeating the task but selecting the option and then clicking Save configuration.

Enabling a cookie compliance banner

You can add a cookie compliance banner to your Developer Portal site by configuring the EU Cookie Compliance module.

Before you begin

You must have administrator access to complete this task.

About this task

The EU Cookie Compliance module provides a customizable pop-up notification that informs visitors to your site that cookies are being used. The notification enables your site visitors to find out more about cookies, and to decide not to browse the site if they disagree with using cookies. You can also provide a link from your cookie compliance banner to an internal or external privacy policy. For more information about configuring an internal privacy policy, see Customizing the privacy policy statement.
The EU Cookie Compliance module is enabled by default. The following instructions show you how to configure the notification.

Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Configuration > System > EU Cookie Compliance.
   The Settings tab of the EU Cookie Compliance page is displayed.
4. In the SETTINGS section select Enable banner, otherwise the notification does not display.
5. In the PRIVACY POLICY section, edit the Privacy policy link field to provide a link to your privacy policy or other page that explains cookies to your site visitors. The default link is `/privacy`, which links to the Privacy Policy content that is configured within the Developer Portal. For more information, see Customizing the privacy policy statement.
   External links should start with `http://` or `https://`. For example:

   `https://example.com/privacy`
6. Configure the remaining options as required. For example, you can change the wording, and edit the display size and color.
7. Click Save configuration to save your updates.

Results

You successfully enabled a cookie compliance banner on your Developer Portal site. New visitors to your site are now prompted to accept cookies.

Related tasks

- Customizing the terms of use statement
- Customizing the privacy policy statement

Enabling code languages for code snippets

You can specify the languages that code snippets for APIs can be displayed in.

Before you begin

You must have administrator access to complete this task.

Procedure

1. On the administrator dashboard, click Configuration > System > IBM API Connect.
2. In the API Code Snippets section, select the check boxes for the languages that you want to make available for the code snippets to be displayed in.
3. Click Save configuration.
   The languages that you have enabled are displayed as options for your code snippets.

Enabling code snippets for SOAP APIs

By default, code snippets are only shown for REST APIs. You can enable code snippets for SOAP APIs.

Before you begin

You must have administrator access to complete this task.

About this task

Code snippets for SOAP APIs use raw HTTP, and do not use any SOAP libraries.

Procedure

1. On the administrator dashboard, click Configuration > System > IBM API Connect.
2. In the API Code Snippets section, select the Display code snippets for SOAP APIs as well as REST APIs check box.
3. Click Save configuration.
   The languages that you have enabled are displayed as options for your code snippets.

Forcing new users to accept terms and conditions

You can extend your Developer Portal site by forcing new users to accept the terms and conditions before they are able to create accounts.

Before you begin

You must have administrator access to complete this task.

Note: From IBM® API Connect Version 10.0.4.0, the **`Legal`** module is included in the list of core Developer Portal modules. The **`Terms of use`** module is still available in the Developer Portal, but the **`Legal`** module provides more configuration options.

About this task

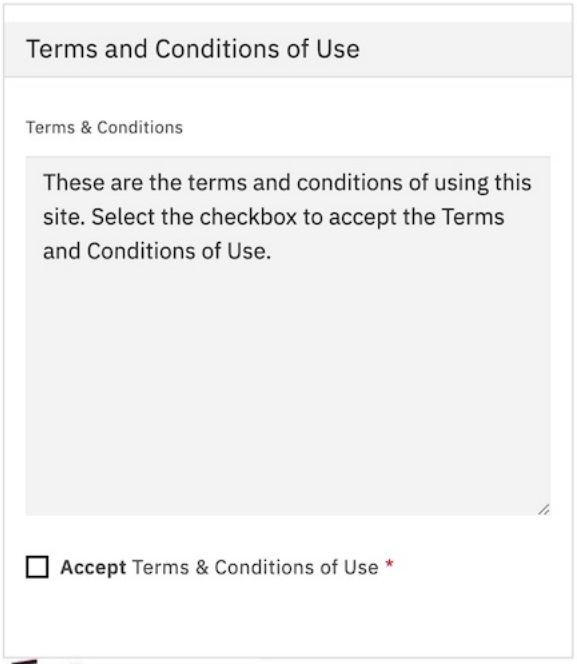
Use the following instructions to enable and configure the **`Legal`** module for your site terms and conditions.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected check box. The disabled modules are displayed without a selected check box.
3. Enter `legal` into the search filter.
   The Legal content item is displayed in the list of content.
4. Select the check box for the **`Legal`** module, and click Enable.
5. Navigate to Configuration > People > Legal.
6. Enter your Terms & Conditions statement, and select the Text format that you require.
7. Click Save configuration.

Results

You successfully enabled the **`Legal`** module. Now, when a user tries to create an account on your Developer Portal site, they must agree to the terms and conditions as a part of the registration.



What to do next

You can edit the terms and conditions page by navigating to Configuration > People > Legal, and making the changes that you require. You can disable the use of terms and conditions during site registration by deleting the terms and conditions content, and then disabling the **`Legal`** module.

If you prefer to use the **`Terms of use`** module, you can enable the module, and then navigate to Configuration > People > Terms of use, and select the page that contains your site terms and conditions, for example **`Terms of use (1)`**. For information about how to customize the statement, see Customizing the terms of use statement. The terms are displayed when you create an account and also from the link in the site footer. You can disable the use of terms and conditions during site registration by disabling the module.

Related information

- Drupal Legal module

Hiding the admin registry on the login form

You can hide the admin registry as an option on the login form. The use case for this option might be a public facing Developer Portal where you want to remove the admin access from customer view.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
3. In the Configuration section, select the check box for **`Hide the admin registry on the login form`**.
4. Click Save configuration.
   The **`Hide the admin registry on the login form`** configuration is saved.

What to do next

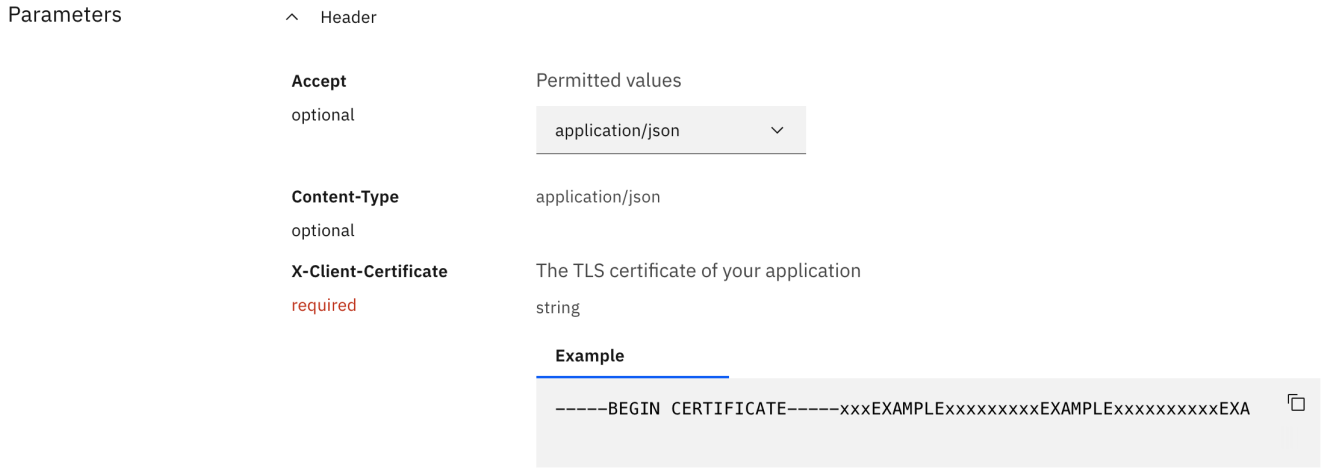
The admin registry is no longer shown on the login form. However, it is still accessible directly. The user can navigate to **`<site_url>/user/login?registry_url=/admin`**.

Hiding the certificate in the header for APIs secured with mutual TLS

You can hide the certificate in the header for APIs secured with mutual TLS. By default, the **`x-client-certificate`** header is shown in an API when mutual TLS is configured. You can choose to turn off this option in your Developer Portal.

Before you begin

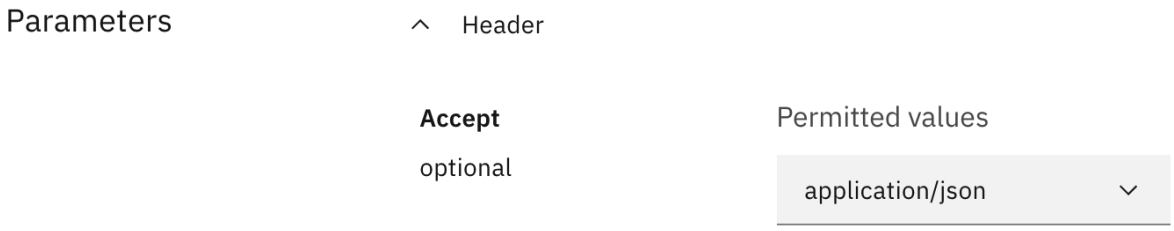
The parameters for the example API showing the certificate.



You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. On the administrator dashboard, click Configuration > System > IBM API Developer Portal.
3. In the Configuration section, deselect the checkbox for **`Show certificate in header for APIs secured with mutual TLS.`**
4. Click Save configuration.
   The **`Show certificate in header for APIs secured with mutual TLS.`** configuration is saved.
   The parameters for the example API now not showing the certificate:



Importing and exporting taxonomies

You can import and export taxonomies in the Developer Portal.

Before you begin

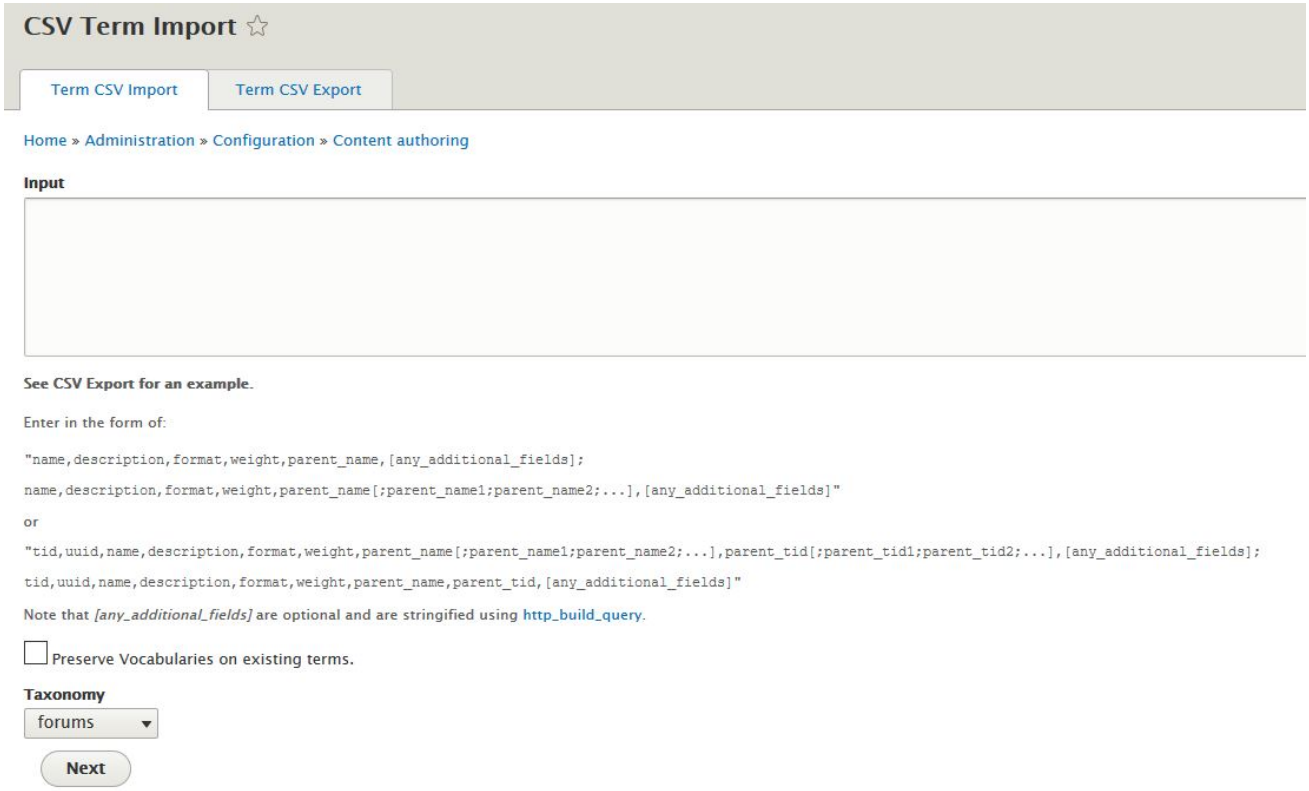
You must have administrator access to complete this task.

Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Configuration > Content authoring > Term CSV Export/Import.
4. Select Term CSV Import, or Term CSV Export.

For Term CSV Import

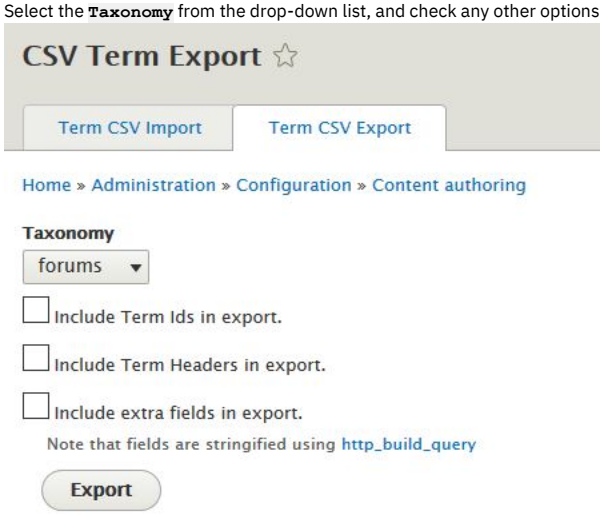
5. Enter your CSV input in the format given in the examples, and select the **`Taxonomy`** from the drop-down list.



6. Click Next.
7. Click Import.

For Term CSV Export

8. Select the **Taxonomy** from the drop-down list, and check any other options that you want.



9. Click Export.

Results

You successfully imported or exported a taxonomy.

Restricting access by IP address

You can restrict access to a role or a user by IP address in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

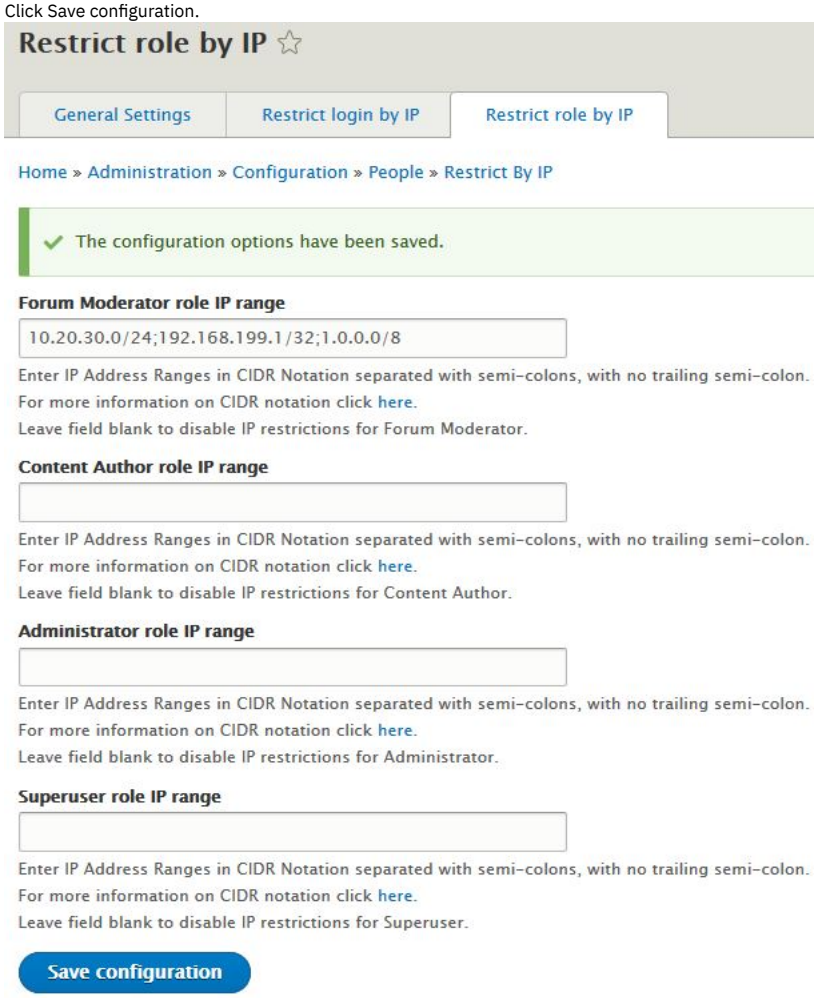
About this task

By restricting access to a role by IP address, that role is unavailable to users outside of the IP address ranges that you define. If a role restriction is triggered, the user's session is unaffected, but the restricted role is no longer available to the user. Role restriction affects only the availability of the restricted role to users. Role restrictions are available for all roles, except `anonymous user` and `authenticated user`.

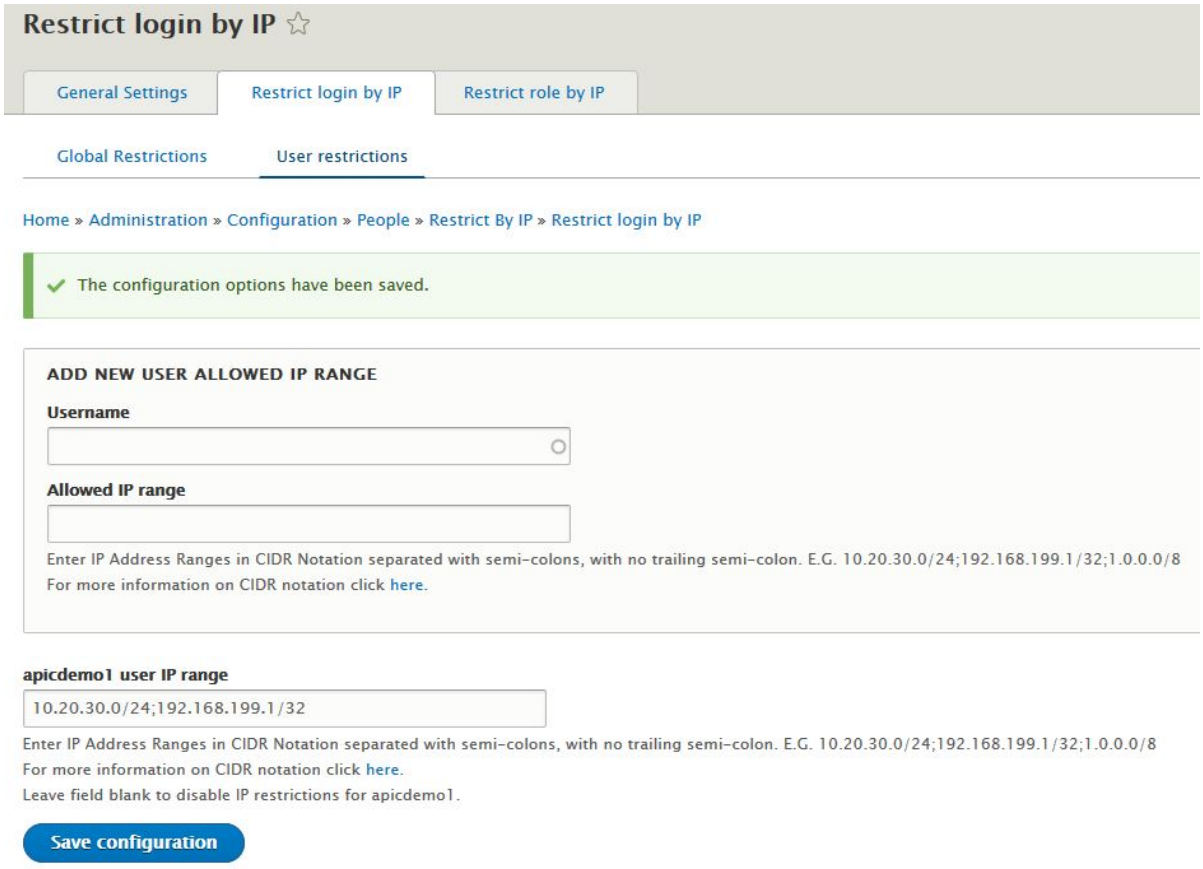
By restricting login access of a user by IP address, the user is unable to log in outside of the IP address ranges that you define. You can also specify global IP address ranges, which apply to all users. IP restrictions are checked on every page load. If a user restriction is triggered by an attempt at logging in being denied, then the user is logged out and sent to the 'error page' that is specified by the site administrator.
Note: IP address ranges must be entered in CIDR notation that is separated with semi-colons and no trailing semi-colon. For more information on CIDR notation, see CIDR format.

Procedure

1. In the Developer Portal, click Configuration > People > Restrict by IP, then click **`Restrict log in by IP`**.
2. On the **`Global restrictions`** tab, enter the address of the page to which the user is redirected to if they are not allowed to log in, in the Login denied error page.
3. Select one of the following options:
   - To restrict access to a role by IP address, complete the following steps:
     - Click Restrict role by IP.
     - Decide which role you want to restrict, the roles that you can restrict include Administrator, Content Author, and Forum Moderator.
     - Enter the IP address range (in CIDR notation) that you do not want to restrict log in for in the field for that role, for example, Forum Moderator role IP range.
     - Click Save configuration.



   - To restrict login access of a user by IP address, complete the following steps:
     - Click Restrict login by IP, then click **`User restrictions`**.
     - In the **`ADD NEW USER ALLOWED IP RANGE`** section, enter a user name in the **`Username`** field.
     - Enter an IP address Range (in CIDR notation), that you do not want to restrict log in for in the **`Allowed IP range`** field.
     - Click Save configuration. Your new log in restriction can be seen after the **`ADD NEW USER ALLOWED IP RANGE`** section.



- To restrict login for all users, complete the following steps:
  - Click Restrict login by IP, then click **`Global restrictions tab`**.
  - Enter the global IP address ranges (in CIDR notation) in the Restrict global login to allowed IP range field.
  - Click Save configuration.

4. To remove an IP restriction, delete the value that is associated with the restriction, then click Save configuration.

Restricting or preventing access from search engines

You can restrict or prevent access to your Developer Portal from search engines. You might want to control the access if you have a development or test site.

Before you begin

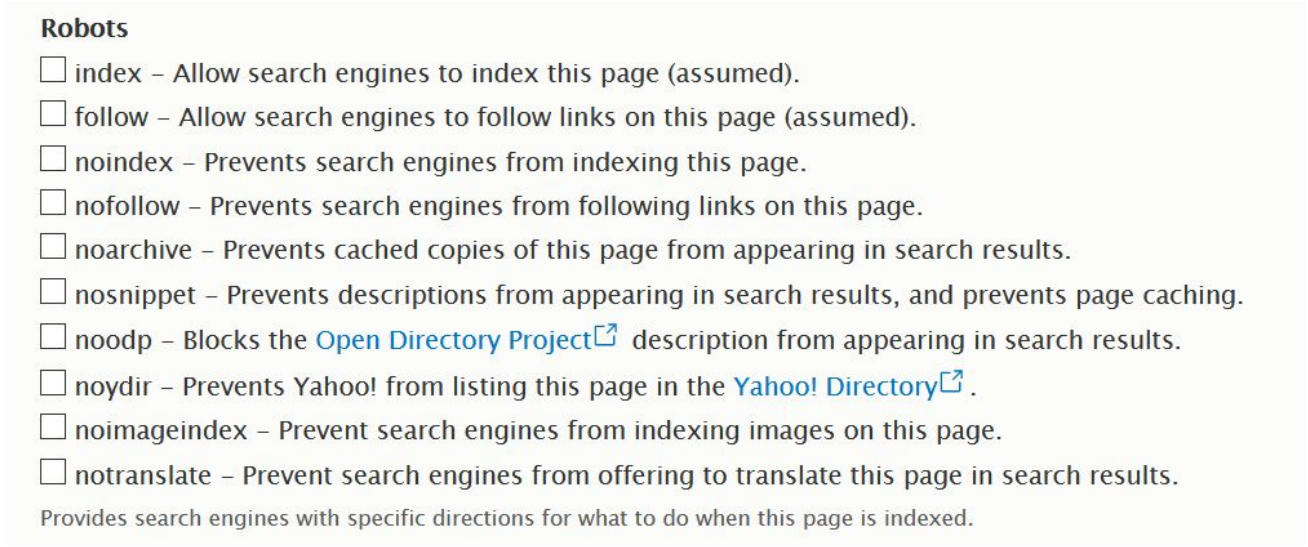
You must have administrator access to complete this task.

About this task

You might have a development or test Developer Portal site on the internet, but do not want them to be indexed by Google, or other search engines, making them visible to your customers. Google explicitly advises not to use **`robots.txt`** as a blocking mechanism. A correct solution is to control access by using Metatag.

Procedure

1. In the Developer Portal, click Configuration > Search and metadata > Metatag.
2. On the **`Global restrictions`** tab, click Edit.
3. In the Advanced section, select the meta tags that you want to use to restrict or prevent the access:



4. Click Save.

Toggling the site in and out of maintenance mode

You can put the Developer Portal site into maintenance mode for short periods of time. During maintenance mode, only an administrator is able to access the site, and all other users who enter the site URL get a maintenance message set by the administrator.

Before you begin

You must have administrator access to complete this task.

Important:

- Maintenance mode is designed for short-term site maintenance; it is not meant for long-term usage. While a site is in maintenance mode, the database is not updated with new content from API Manager. As soon as the maintenance tasks are finished, you must take the site out of maintenance mode.
- Do not log out of the Developer Portal while the site is in maintenance mode, as you will not be able to log back in to the UI. In this case, maintenance mode can be turned off only by using the toolkit CLI; see the following instructions for details.

Procedure

To put the site into maintenance mode, complete the following steps.

1. Click Configuration in the administrator dashboard.
2. Under the DEVELOPMENT heading, click Maintenance mode.
3. Select the Put site into maintenance mode checkbox.
4. Enter a site message, or leave the default site maintenance text.
   This message is what will be seen by users who visit the site while it is in maintenance mode.
5. Click Save configuration.
   Your site is now in maintenance mode.

To take the site out of maintenance mode, complete one of the following steps.

6. If you remained logged in to the site as the admin user during the maintenance period, you can use the Developer Portal UI to disable maintenance mode:
   a. Click Configuration in the administrator dashboard.
   b. Under the DEVELOPMENT heading, click Maintenance mode.
   c. Clear the Put site into maintenance mode checkbox .
      Your site is now available to users, and out of maintenance mode.
7. If you're no longer logged in to the site, you can use the toolkit CLI to disable maintenance mode:
   a. Log in to the toolkit CLI as a Provider org:

      ```
      apic login --server management_server --realm provider/identity_provider --username provider_username --password
      provider_password
      ```

      You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

      ```
      apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
      ```

      For example,

      ```
      apic identity-providers:list --scope provider --server myserver.com --fields title,realm
      total_results: 2
      results:
        - title: API Manager User Registry
          realm: provider/default-idp-2
      ```

```
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

b. Run the following command to disable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --
state_key system.maintenance_mode  --input_format integer --state_value 0
```

Where:

- *`management_server`* is the endpoint URL of your management server (required).
- *`catalogid/name`* is the ID or name of the catalog that your site belongs to (required).
- *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
- `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
- `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
- `--state_value` is the value to assign to the state key (required). Set to `0` to disable maintenance mode.

For example, to disable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key
system.maintenance_mode  --input_format integer --state_value 0
Successfully set state system.maintenance_mode to the value of 0.
```

Your site is now available to users, and out of maintenance mode.

What to do next

You can use the toolkit CLI to enable, and disable, maintenance mode. For more information, see the scenario How to enable and disable maintenance mode on your Developer Portal.

Related tasks

- Getting started with the Portal CLI commands

Related reference

- Using the drupal-state commands

Viewing available updates

You can view available updates in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To view available updates, complete the following steps:

1. Click Reports in the administrator dashboard.
2. Click Available updates.
3. You can see what updates are available from this view. However, updates can be applied only by applying the latest available API Connect release; see Upgrading in a Kubernetes environment, or Upgrading in a VMware environment, for more information.

Managing Developer Portal security

You can manage multiple security elements in the Developer Portal.

- **Configuring CAPTCHA**
  You can add a CAPTCHA challenge to any of your Developer Portal pages to protect your site from spam.
- **Configuring reCAPTCHA**
  You can configure reCAPTCHA on your Developer Portal site to provide more protection from spam and abuse.
- **Configuring session timeout and limit**
  You can specify session timeout settings to control when a user is automatically logged out of the Developer Portal. You can also restrict the number of sessions a single user has available to them.
- **Configuring the timeout length for the password reset link**
  You can configure the timeout length for the one-time password reset link that is sent from the Developer Portal to the admin user account.
- **Configuring your site password policy**
  You can configure your site password policy to define the constraints that are applied to the passwords of your Developer Portal users.

- **Disabling CORS warnings**
  You can disable cross-origin resource sharing (CORS) warnings for unenforced APIs in the Developer Portal.
- **Disabling live testing of APIs**
  You can disable live testing of APIs in the Developer Portal to restrict exposure of an API.
- **How to enable penetration testing, and information about Developer Portal cookies**
  Many security features are available in the Developer Portal, but if you need to do a penetration test of your Developer Portal site, some of these features might block the scan that is required as part of the testing. This topic provides guidance on the security features that you might need to disable to allow penetration testing of your site, as well as information about Developer Portal cookies.
- **How to manage IP security in the Developer Portal**
  The Developer Portal offers the ability to perform various IP address security measures, such as adding and removing specific IP addresses from the banned IP address list, automatically banning client IP addresses by using the Drupal Perimeter Defence module, or managing login security by using flood control.
- **Managing banned IP addresses**
  You can ban specific IP addresses from accessing your Developer Portal site.
- **Using flood control for login security**
  You can configure login security for your Developer Portal by using flood control.
- **Login security**
  You can configure the login security for your Developer Portal at an IP and user level. You can also configure login security for Developer Portal contact forms.
- **Using the Security kit**
  You can improve the security of your website by configuring various options that are available in the Security Kit module, in the Developer Portal.
- **Using Honeypot for spam protection**
  Honeypot protection provides security mechanisms to protect your Developer Portal site from form submission by spam bots. If spam bot activity is detected, form submission is blocked.

Configuring CAPTCHA

You can add a CAPTCHA challenge to any of your Developer Portal pages to protect your site from spam.

Before you begin

You must have administrator access to complete this task.

About this task

The types of CAPTCHA challenges that are available include Image and Math based CAPTCHAs. Both types of CAPTCHA challenges are configurable in the CAPTCHA dialog box. By default, your Developer Portal site is configured with the Image based CAPTCHA challenge enabled.
You can also configure reCAPTCHA on your site. For more information, see Configuring reCAPTCHA.

Procedure

To configure CAPTCHAS, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click CAPTCHA module settings.
   The CAPTCHA settings window opens.
4. Use the CAPTCHA configuration options provided as required. For example, you can complete the following tasks:
   a. Specify the CAPTCHA challenge type.
   b. Specify which forms must include a CAPTCHA challenge.
   c. Add a description to the CAPTCHA.
   d. Enable CAPTCHA statistics.
5. Click Save configuration to save your changes.

Configuring reCAPTCHA

You can configure reCAPTCHA on your Developer Portal site to provide more protection from spam and abuse.

Before you begin

You must have administrator access to complete this task.

About this task

reCAPTCHA is a CAPTCHA-like system that is designed to establish that a computer user is human. The types of reCAPTCHA that are available include Checkbox, Invisible, and Android.
Note: To use reCAPTCHA the administrator of your Developer Portal is required to sign up to Google's reCAPTCHA API, and your portal server must have internet access.

Procedure

To configure reCAPTCHA, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.

2. Click Configuration > People > CAPTCHA module settings > reCAPTCHA in the administrator dashboard.
   The reCAPTCHA settings window opens.
3. In the GENERAL SETTINGS section, click register for reCAPTCHA.
   The Google registration site for reCAPTCHA is displayed.
4. Follow the instructions on Google to register your Developer Portal site, and obtain the Site key and Secret key.
5. Return to the reCAPTCHA settings window in the Developer Portal, and enter the Site key and Secret key that you obtained from Google into the GENERAL SETTINGS section.
6. Optional: Update the WIDGET SETTINGS section as required.
7. Click Save configuration to save your changes.
8. Click the CAPTCHA Settings tab, and in the FORM PROTECTION section change the Default challenge type to reCAPTCHA (from module reCAPTCHA).
9. Click Save configuration.

Results

The reCAPTCHA module is now enabled on your Developer Portal site.

Related tasks

- Configuring CAPTCHA

Configuring session timeout and limit

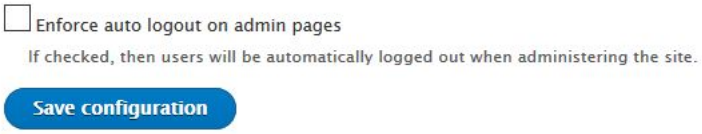
You can specify session timeout settings to control when a user is automatically logged out of the Developer Portal. You can also restrict the number of sessions a single user has available to them.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. To manage the automated logout settings, complete the following steps:
   a. Click Configuration in the administrator dashboard.
   b. In the PEOPLE section, click Automated logout settings.
      Note: You must check **`Enforce auto logout on admin pages`** for **`autologout`** to apply when navigating the admin areas of the site.

   c. Use the session timeout configuration options provided as required. For example, you can complete the following tasks:
      - Specify the length of inactivity time, in seconds, before a user is automatically logged out.
      - Specify session timeout values for individual user roles.
        Note: The default session timeout is 30 minutes.
      - Provide a redirection URL at logout.
      - Provide session expiry message texts.
   d. Click Save configuration to save your changes.
3. To restrict the number of sessions available to a user, complete the following steps:
   a. Click Configuration in the administrator dashboard.
   b. In the PEOPLE section, click Session limit.
   c. Specify the default maximum number of sessions for a user by entering the number into the Default maximum number of active sessions field.
   d. Select the required action from the When the session limit is exceeded options.
   e. From the Logged out message severity list, select the severity of the message that the user receives when they are logged out.
   f. Optionally, in the ROLE LIMITS section, specify separate session limits for each role.
   g. Click Save configuration to save your changes.

Configuring the timeout length for the password reset link

You can configure the timeout length for the one-time password reset link that is sent from the Developer Portal to the admin user account.

Before you begin

You must have administrator access to complete this task.

About this task

By completing the steps in this task, you are configuring the timeout length for the one-time password reset link that is sent to the admin user account by the Developer Portal.

Note:

- Setting the password reset timeout length in the Developer Portal affects only the admin user account.
- You are not affecting the user activation or password reset links that are used by any other identification provider type in API Manager.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click Account settings.
   The Account settings window opens.
4. In the Account settings window, navigate to the PASSWORD RESET TIMEOUT section.
5. Enter your required value for the timeout in the Password Reset Timeout field.
   Note: The timeout value is specified in seconds, and is 86400 seconds by default.
6. Click Save configuration.
   You have configured the timeout length for the one-time link that is sent from the Developer Portal for the admin user account.

Configuring your site password policy

You can configure your site password policy to define the constraints that are applied to the passwords of your Developer Portal users.

Before you begin

You must have administrator access to complete this task.

About this task

Any passwords that are stored in the API Manager must comply with the API Manager password policy. You can configure your password policy to be the same as or stricter than the API Manager password policy.

Procedure

To configure your site password policy, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SECURITY section, click Password Policy.
   The Password Policies page opens.
4. In the OPERATIONS column for the default password policy, click Edit.
   Note: You can create a new password policy by clicking Add Policy; this allows you to apply different password policy settings to different roles. If you need only a single password policy, just configure the pre-supplied default policy.
5. Click Next.
   The Configure Constraints page opens.
6. Configure the password constraints as required.
   a. To configure a constraint, click Edit in the OPERATIONS column for the required constraint.
   b. To delete a constraint, select Delete in the OPERATIONS column.
   c. To add a constraint, select the required constraint type from the drop-down list, then click Configure Constraint Settings.
   d. When you are done, click Next.
      The Apply to Roles page opens.
7. Select the roles to which the password policy settings are to be applied. then click Finish.

Disabling CORS warnings

You can disable cross-origin resource sharing (CORS) warnings for unenforced APIs in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

By disabling the CORS warnings, users will not receive further warnings regarding the usage of unenforced APIs.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SYSTEM section, click IBM API Connect.
4. Ensure that the Display CORS warnings for unenforced APIs check box is cleared.
5. Click Save Configuration.

You have disabled CORS warnings in the Developer Portal. To enable CORS warnings, repeat the steps and ensure that the Display CORS warnings for unenforced APIs check box is selected.

Disabling live testing of APIs

You can disable live testing of APIs in the Developer Portal to restrict exposure of an API.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SYSTEM section, click IBM API Connect.
4. Ensure that the Allow live testing of APIs check box is cleared.
5. Click Save configuration.
   You have disabled live testing of APIs. To enable live testing of APIs, repeat the steps and ensure that the Allow live testing of APIs check box is selected.

How to enable penetration testing, and information about Developer Portal cookies

Many security features are available in the Developer Portal, but if you need to do a penetration test of your Developer Portal site, some of these features might block the scan that is required as part of the testing. This topic provides guidance on the security features that you might need to disable to allow penetration testing of your site, as well as information about Developer Portal cookies.

Penetration testing, or pen testing, is the practice of testing a computer system, network, or web application for potential security vulnerabilities that an attacker might exploit. For certain elements of your pen testing, you might want to disable some of the security features that block scanning of the Developer Portal. The following sections describe the security features in the Developer Portal that you might need to disable. The actual requirements depend on your Developer Portal configuration, and on the type of pen testing that you are running. Note also the final section, which discusses testing for Distributed Denial of Service attacks.
Important:

- If you run a pen test, and you give the automated test tool the admin credentials, the tool submits all of the administration configuration forms as well, and this action is likely to break the Portal site configuration. Best practice is to use different credentials for the testing.
- If you disable security modules, and then run security scans, your Portal site configuration might be broken by the scanning process. Before running any security testing, ensure that you have a valid backup of your Portal site.
- If your Portal site configuration is broken by the security testing, then you must delete the Portal site from the API Manager Catalog settings and re-create it, or restore it from a backup.
  For information about restoring your site, see the appropriate topic for your environment:
  - Restoring the Developer Portal on Kubernetes and OpenShift
  - Restoring the Developer Portal on VMware
- Note that disabling of security modules makes it highly likely that you will find issues that are not actually vulnerabilities, because you have disabled the modules that are designed to protect against these vulnerabilities. Therefore, any issue that is found in these circumstances must be replicated in an environment without the security modules disabled, before contacting IBM Support.
- Do not use the **`Enforced`** mode of the **`Content Security Policy`**, this mode is not supported and can break core Developer Portal functionality such as API rendering and the content editor. The **`Content Security Policy`** must be in **`Report only`** mode.

Note: You might need to disable the **`Honeypot`** module to allow your penetration test to run. For more information, see Using Honeypot for spam protection.
This topic contains the following sections:

- CAPTCHA and reCAPTCHA verifications
- Drupal Perimeter Defense module
- Login Security module
- Developer Portal cookies
- Should I test the Developer Portal for vulnerability to Distributed Denial of Service attacks?

CAPTCHA and reCAPTCHA verifications

CAPTCHA and reCAPTCHA verifications are a type of challenge-response test that are used in web applications to check whether the user is human. By default, your Developer Portal site is configured with an Image based CAPTURE challenge enabled, but you can also configure a Math based CAPTCHA challenge, and Checkbox, Invisible, and Android reCAPTCHA challenges. If you need to disable these verifications before pen testing, see the following instructions.

To disable the CAPTCHA and reCAPTCHA verifications, you must disable the modules as follows:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Extend > Disable module.
4. Search through the list of modules, and select the check boxes next to the Image CAPTCHA and reCAPTCHA modules.
5. Click Disable, and click Disable again to confirm.
6. On the Disable tab, select the check box next to the CAPTCHA module, and click Disable. The display shows the forms that the CAPTCHA configuration is removed from.
7. Click Disable again to confirm. Note that the modules must be disabled in this order.

The CAPTCHA and reCAPTCHA modules are now disabled, and you can complete your pen testing. After pen testing, you can enable the modules again by clicking Extend on the administrator dashboard, selecting the check box for each module in the List tab, and clicking Enable.

Drupal Perimeter Defense module

The Drupal Perimeter Defense module (machine name: perimeter) immediately bans the IP addresses of those users that send suspicious requests to your site, so any pen testing scans would be blocked. This module is enabled by default on the Developer Portal. For more information about the module, see the Drupal documentation Drupal Perimeter Defense module.

If you need to disable the Drupal Perimeter Defense module before pen testing, see the following instructions:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Extend > Disable module.
4. Search through the list of modules, and select the check box next to the Drupal Perimeter Defense module.
5. Click Disable, and click Disable again to confirm.

The Drupal Perimeter Defense module is now disabled, and you can complete your pen testing. After pen testing, you can enable the module again by clicking Extend on the administrator dashboard, selecting the check box next to the Drupal Perimeter Defense module in the List tab, and clicking Enable.

Login Security module

The Login Security module (machine name: login_security) enables the site administrator to improve the security options in the login operation of your Developer Portal site. For example, controlling the number of invalid login attempts before blocking users, and denying access by IP address. This module is enabled by default on the Developer Portal. For more information about the module, see the Drupal documentation Login Security module.

If you need to disable these security options before pen testing, you need to manually modify the configuration of your security options according to your requirements. For example, you might need to change the number of login attempts that are allowed before a user or IP address is blocked. To configure the security options, see the following instructions:

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. On the administrator dashboard, click Configuration > People > Login Security.
4. In the Login Security window, update the security options according to your requirements.
5. Click Save configuration to save your updates.

The Login Security options are now updated, and you can complete your pen testing. After pen testing, you can reconfigure your security options in the Login Security window.

Developer Portal cookies

Cookies are small text files that are placed on a computer by websites when they are visited. The Developer Portal uses cookie that remain on the computer for varying times. Some cookies expire at the end of each session, and some remain longer to enhance the user experience in the future.
Note: The cookie generated by Bigpipe does not have the **`httponly`** flag set. Thus, JavaScript can delete the cookie whenever required.
The list here explains the cookies that are used by the Developer Portal, and why they are used.
Note: Any load balancer that sits in front of the Developer Portal must be a transparent proxy that enables all cookies returned by the Developer Portal to pass through to the browser.

Drupal session identifier cookie
: This is the main cookie that all of the Developer Portal sites have, and it stores the session identification. The session cookie is named in the format SSESS*UUID*, where *UUID* is a randomly generated unique string of characters for that session, and this is the session token itself. The session cookie is always marked with the following flags as instructions to the browser client:

  - **`Secure`** - means that the cookie is sent only over a secure connection (that is, over TLS).
  - **`HttpOnly`** - means that the browser adds this cookie on outbound requests only, and not provide the value to client side javascript.

session_store.id cookie
: This cookie is used to store information on where a user is in the subscription flow, and any options that they selected, if they haven't yet signed in, so the Developer Portal can redirect them back there later. The session_store.id cookie is also marked with the **`Secure`** and **`HttpOnly`** flags, and it does not contain any confidential information.

Drupal.visitor.startSubscriptionWizard cookie
: This cookie is used to store information on where a user started in the subscription flow if they haven't yet signed in, so the Developer Portal can redirect them back there later. The Drupal.visitor.startSubscriptionWizard cookie is also marked with the **`Secure`** and **`HttpOnly`** flags, and it does not contain any confidential information.

portal_web cookie
: This cookie is used to achieve session persistence, which it does by ensuring that web traffic between the user and the Developer Portal is correctly routed to the server. The portal_web cookie is used for both HTTP and HTTPS communications, so that routing is meaningful irrespective of the transfer protocol, and ensuring consistent behavior if the switching of protocols is required. The cookie contains only a hash of the pod that served the traffic to maintain session persistence, and does not contain any confidential information. The cookie is provided by the Portal ingress. This cookie is not marked as **`Secure`** or **`HttpOnly`**, as the Kubernetes ingress does not yet support these attributes.

Other cookies
: In addition to the cookies listed previously, there are other (usually temporary) cookies that might be created depending on the configuration options of your Developer Portal site. For example, if the site uses OAuth for authentication, then during the OAuth token activation flow a JSON Web Token is sent to the browser. This token is stored in a cookie for a limited lifetime. This action is part of the normal function of an OAuth flow. As the creation of these other cookies depends on your particular Developer Portal configuration, we cannot provide a complete list of cookies here.

Should I test the Developer Portal for vulnerability to Distributed Denial of Service attacks?

When you run penetration testing, it's important to understand that individual systems are not responsible for (nor capable of dealing with) Distributed Denial of Service (DDoS) attacks. DDoS attacks are launched by huge numbers of external computers simultaneously, with a goal of using up all of the resources of a system, and thereby denying service to legitimate users. These kinds of attacks are usually defended against at the level of the network and edge gateways, and information about how to protect against them is out of the scope of this topic. Although the Developer Portal does have some capabilities, including the modules mentioned previously, for potentially mitigating some of the effects of a DDoS attack, do not rely on these capabilities as the first line of defense; rather they are designed to assist in dealing only with any elements of an attack that might have gotten through the outer network layers of protection.

Related tasks

- Configuring CAPTCHA
- Configuring reCAPTCHA
- Disabling modules
- Login security
- Blocking and unblocking specific users
- Managing banned IP addresses

Related information

- Understanding Drupal

How to manage IP security in the Developer Portal

The Developer Portal offers the ability to perform various IP address security measures, such as adding and removing specific IP addresses from the banned IP address list, automatically banning client IP addresses by using the Drupal Perimeter Defence module, or managing login security by using flood control.

See the following links for information about how to manage IP security in the Developer Portal:

- Enable and disable IP security
- Correctly passing through client IP addresses
- Managing banned IP addresses

Enable and disable IP security

The ability to enable or disable IP security related actions can be controlled at the Portal service level by using the Developer Portal CLI command **`ip-security-enabled`**. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. When IP security is disabled, all IP related security is switched off. For example, the perimeter module won't block client IP addresses when IP security is disabled. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. For more information, see Using the ip-security-enabled command.

Correctly passing through client IP addresses

In order to correctly use modules that make use of client IP banning, such as the Drupal Perimeter Defence module, you must ensure that any external load balancer that fronts the Portal cluster passes through the client IP address. This can be achieved by passing the client IP address through in an **`'x-forwarded-for'`** header, or by making use of the proxy protocol (provided both the load balancer and the ingress controller are compatible with the proxy protocol, and have the protocol enabled). Failure to correctly pass through the client IP address, results in the load balancer IP address being blocked when a client attempts to send a suspicious request to the portal.

If you are fronting the Portal cluster with an HAProxy setup acting as the load balancer, then you can make use of the proxy protocol by adding the **`send-proxy`** directive to the end of the Portal server declarations in the HAProxy configuration file, for example:

```
server portal0 portal_host:port check send-proxy
```

You must restart the HAProxy for the change to take effect. Note that if your load balancer IP address has already been blocked, you will need to remove the blocked IP address from the banned list by using the **`security`** command in theDeveloper Portal CLI, see Using the security command.

If you find your load balancers are being banned by Portal's IP Security modules, then you might need to configure your system's ingresses to send the **`'x-forwarded-for'`** headers. For instance, the default behavior of the **`nginx-ingress-controller`** is to ignore the inbound **`'x-forwarded-for'`** header and construct a new one. To configure the **`nginx-ingress-controller`** to pass through the inbound **`'x-forwarded-for'`** header you need to add the following to the **`nginx-ingress-controller`** config map:

```
data:
 compute-full-forwarded-for: "true"
 use-forwarded-headers: "true"
```

For more information about the Drupal Perimeter Defence module, see Drupal Perimeter Defense module.

Managing banned IP addresses

You can manage banned IP addresses for a particular site by using the administrator dashboard for a particular Developer Portal site. For more information, see Managing banned IP addresses.

Related concepts

- How to enable penetration testing, and information about Developer Portal cookies

Related tasks

- Disabling modules
- Login security
- Blocking and unblocking specific users
- Managing banned IP addresses

Managing banned IP addresses

You can ban specific IP addresses from accessing your Developer Portal site.

Before you begin

You must have administrator access to complete this task.

Procedure

To manage banned IP addresses, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click IP address bans.
   The IP address bans window opens.
4. You can complete the following tasks:
   a. To ban an IP address, enter the address in the IP address field, then click Add.
      The address is added to the BANNED IP ADDRESSES list.
   b. To remove an IP address from the banned list, click Delete alongside the required address.

   Note:
   - There is automatic login "flood" protection built into the Developer Portal. This feature means that an excess of login attempts from the same IP address in one hour will cause that IP to be temporarily banned. For more information, see Using flood control for login security.
   - You can also clear banned user/IP addresses by using the Developer Portal CLI command **`security:clear-bans`**. For more information, see Using the security command.

Using flood control for login security

You can configure login security for your Developer Portal by using flood control.

Before you begin

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to Configuration > System > Flood Control.
3. Enter values in the following fields in the LOGIN SETTINGS section according to your requirements:
   - Failed IP login limit (min 1) - the number of failed login attempts from the same client IP.
   - Failed IP login window in seconds (0 = Off) - the time window, in seconds, for the number of failed login attempts from the same client IP.
   - Failed User login limit (min 1) - the number of failed login attempts from a user.
   - Failed User login window in seconds (0 = Off) - the time window, in seconds, for the number of failed login attempts from a user.
4. Enter values in the following fields in the CONTACT FORM section according to your requirements:
   - Emails sent limit - the number of emails that can be sent.
   - Emails sent window in seconds - the time window, in seconds, for the number of emails that can be sent.
5. Click Save configuration.

What to do next

To unblock a user, see Blocking and unblocking specific users. To unblock an IP address, see Managing banned IP addresses.

Login security

You can configure the login security for your Developer Portal at an IP and user level. You can also configure login security for Developer Portal contact forms.

Before you begin

Important: The Login Security module is being removed from future releases. It has been replaced by Flood Control. Existing users of the Login security module should configure Flood Control with the same limits and time windows as specified here. For more information, see Using flood control for login security.

You must have administrator access to complete this task.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the PEOPLE section, click Login Security.
4. Enter values in the following fields in the GENERAL SETTINGS section according to your requirements:
   - Track time - the time window to check for security violations.
   - User - the number of login failures before blocking a user.
   - Soft host - the number of login failures before soft blocking a host. The host can still browse the site as an anonymous user.
   - Host - the number of login failures before hard blocking a host. The host cannot access the site at all.
   - Attack detection - the number of login failures before warning about an ongoing attack.
5. Select the check boxes that satisfy your notification requirements in the NOTIFICATION section:
   - Disable login failure error message
   - Notify the user about the number of remaining login attempts
   - Display last login timestamp
   - Display last access timestamp
6. Optional: Edit the EMAIL FOR ONGOING ATTACK DETECTION section.
7. Optional: Edit the EMAIL FOR BLOCKED ACCOUNT section.
8. Optional: You can edit the notification texts that are displayed by configuring the following options:
   - Failed login attempt
   - Banned host (soft IP ban)
   - Banned host (hard IP ban)
   - Blocked user by uid
9. After you have configured your login security options, click Save configuration.

What to do next

To unblock a user, see Blocking and unblocking specific users. To unblock an IP address, see Managing banned IP addresses.

Using the Security kit

You can improve the security of your website by configuring various options that are available in the Security Kit module, in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

The Security Kit module provides your website with various options to mitigate risks of common web application vulnerabilities like Cross-site Scripting, Cross-site Request Forgery and Clickjacking. It also has some options to improve your SSL/TLS security.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the SYSTEM section, click Security Kit settings.
4. In the Security Kit module, configure the security settings for your website as required.
5. When you are done, click Save configuration.

Using Honeypot for spam protection

Honeypot protection provides security mechanisms to protect your Developer Portal site from form submission by spam bots. If spam bot activity is detected, form submission is blocked.

Before you begin

You must have administrator access to complete this task.

About this task

Honeypot protection provides the following security mechanisms:

- A hidden field, unseen by users, is added to the form. If a value has been entered in the field when the form is submitted then this indicates that the form was completed by a spam bot, and the submission is blocked. You can specify the name of the hidden field.
- If the form is submitted before a specified time has elapsed (five seconds by default), it is assumed that this is too short a time for a human to have completed the form, and the submission is blocked. You can specify the time length.

Honeypot protection is provided by the Honeypot module, which is enabled by default.
Note: If you want to carry out automated testing of your Developer Portal, you might need to disable the Honeypot module, because Honeyspot spam protection is designed specifically to block automated Developer Portal usage. For details on how to disable a module, see Disabling modules.

Procedure

To configure Honeypot for spam protection in the Developer Portal, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. In the CONTENT AUTHORING section, click Honeypot configuration.
4. Specify the forms that you want to protect with Honeypot.
    a. To enable Honeypot protection for all the forms on your Developer Portal site, select Protect all forms with Honeypot.
    b. To choose which forms you want to protect with Honeypot, clear the Protect all forms with Honeypot check box, then select the required forms in the HONEYPOT ENABLED FORMS section.

   By default, Honeypot protection is enabled for all user management forms and all comment forms.
5. To have details of all blocked form submissions written to the log file, select Log blocked form submissions.
6. In the Honeypot element name field, specify the name of the hidden form field.
   The default field name is url. You need to change the value if your form already has a field of the same name. For the most effective protection, use a generic field name; for example, email, homepage, or link.
7. In the Honeypot time limit field, specify the number of seconds that must elapse before it is assumed that a form is being submitted by a human rather than a spam bot. If the form is submitted before this time has elapsed then the submission is blocked.
   The default value is five seconds.
8. When done, click Save configuration.

People

You can manage and customize the user experience in the Developer Portal, by altering permissions and assigning roles to specific users.

To learn how to configure the user experience for your Developer Portal site, use the following section links.

- **Blocking and unblocking specific users**
  You can block and unblock specific users in the Developer Portal.
- **Controlling access to Developer Portal content**
  In IBM® API Connect, you can restrict access to content for Consumer organizations through an access content list. However, you can also assign permissions in the Developer Portal that override the access content list, and allow users to access and edit all of the content for Products, APIs, and Applications.
- **Creating a developer account to customize API properties**
  You can allow specific users in the Developer Portal to customize the properties for certain APIs, including APIs that are not viewable by the administrator, or other customizable features, by creating an internal Consumer organization.
- **Email Subscribers**
  As a site administrator, you can use the email subscribers wizard to send messages to your Developer Portal community.
- **How to style the Developer Portal emails**
  You can apply a theme to the HTML emails that are sent by the Developer Portal by using CSS and templating.
- **Working with roles in the Developer Portal**
  As a site administrator, you can create and customize site configuration roles in the Developer Portal.

Blocking and unblocking specific users

You can block and unblock specific users in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To block or unblock a specific user, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
   A list of users is displayed.
3. Select the check box next to the target user.
4. From the drop-down list under Action, select Block the selected user(s) or Unblock the selected user(s).
5. Click Apply to selected items.
   You have blocked or unblocked specific users from the Developer Portal.
   Note: If a user reports that they are unable to login, but they do not appear as blocked, then it might be that they have been blocked by Flood Control. To unblock such users, select Configuration > System > Flood unblock. You can configure your Flood Control settings; see Using flood control for login security for more information.

Related tasks

- Using flood control for login security

Controlling access to Developer Portal content

In IBM® API Connect, you can restrict access to content for Consumer organizations through an access content list. However, you can also assign permissions in the Developer Portal that override the access content list, and allow users to access and edit all of the content for Products, APIs, and Applications.

Before you begin

You must have administrator access to complete this task.

About this task

A user with a Content Author role is a curator of content for the Developer Portal. A Content Author user can perform the following actions:

- Set taxonomies on the content
- Provide custom icons for content
- Upload additional documentation files
- Create documentation pages in the Developer Portal.

By default, Content Author users are assigned the following user permissions:

Edit any Product content
: The user can access and edit all of the Products available in the Catalog that the Developer Portal is associated with. The user can also access APIs that belong to those Products.

Edit any API content
: The user can access and edit all of the APIs that are available in the Catalog, but not the Products with which they are associated. (Note that this permission is a subset of Edit any Product content; as Content Author users are assigned both permissions they can access and edit all of the Products and the associated APIs that are available.)

The Edit any Application permission grants the user access to all of the Applications that are in the Developer Portal, but is not assigned to any user or role by default.
Note: There are security implications if you grant the Edit any Application permission, and it should be assigned only to trusted roles.
A user can configure, attach, and upload content if they are assigned the appropriate permissions.

Procedure

To check and assign permissions that grant users access to all content, proceed with one of the following options:

- Assign a user to the Content Author role. For more information, see Assigning users to a role.
- Create a custom role, to which you can add the relevant permissions and add users to your new custom role. For more information, see Creating a new role, Assigning permissions to a role, and Assigning users to a role.

Creating a developer account to customize API properties

You can allow specific users in the Developer Portal to customize the properties for certain APIs, including APIs that are not viewable by the administrator, or other customizable features, by creating an internal Consumer organization.

Before you begin

You must have administrator access to complete this task.

About this task

Every user in the Developer Portal must be a member of a Consumer organization. To enable additional users to be editors of particular content in the Developer Portal, you can create an internal Consumer organization. You can then invite these additional editors to this organization, and assign them a specific role that grants certain editing permissions. For example, the following API properties can be customized by any member of the new Consumer organization that have been assigned the Administrator or Content Author role:

- Image uploading
- Tag assignment
- File attachment
- Custom field editing

Procedure

To create a new Consumer organization within the Developer Portal, complete the following steps:

1. Log in to the Developer Portal as an administrator or Content Author.
2. Click the arrow next to `your_organization_name` from the Developer Portal home page, and select Create organization.
3. In the Organization Title field, type the name of the organization that you want, then click Submit.

To invite the editor users to your new organization, complete the following steps:

4. Switch to your newly created organization by clicking the arrow next to `your_current_organization_name`, and select your new organization name from the list.
5. Click the arrow next to `your_new_organization_name` and select My organization.
   Your new organization page is displayed.
6. For each new user, click Invite, complete their Email details, and click Submit. Note that you can leave the default role of Developer selected and then change the role later, or you can select the Administrator role now. However, be aware that an administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps. You should be careful to ensure that only trusted users are given this access and level of control of your site. Each new user will receive an invitation email, and they need to click on the link in the email to activate their Developer Portal account.

To assign the required permissions to the users of your organization, you must log in as an administrator and complete the following steps:

7. Log in to the Developer Portal as an administrator.
8. If the administrator dashboard is not displayed, click Manage to display it.
9. Click People on the administrator dashboard.
10. Select the check box for the user, or users, to whom you want to give new permissions.
11. Under the Action heading, select the relevant role from the drop-down list.
    For example, selecting Add the Content Author role to the selected user(s) means they can edit images and content within the Developer Portal.
12. Click Apply to selected items to update the roles.

What to do next

If you want specific Products or APIs to be customized by the editor users, you must ensure that the relevant Products are visible to the internal Consumer organization that you have created. For more information on how to make a plan visible in API Manager, see Changing the availability of a Product.
Note that you can also create new roles and assign specific permissions to those roles. For more information, see Creating a new role and Assigning permissions to a role.

Related concepts

- Working with roles in the Developer Portal

Email Subscribers

As a site administrator, you can use the email subscribers wizard to send messages to your Developer Portal community.

You can email all the subscribers of a specific API, plan, product, or all registered consumer organizations. For each consumer organization, you can select whether to email just the owner or all of the members. Each recipient receives an individual email.

You can use the wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

- **Emailing all subscribers**
  As a site administrator, you can use the email **`All users`** wizard to email all the subscribers of all the consumer organizations on your Developer Portal.
- **Emailing api subscribers**
  As a site administrator, you can use the email **`API subscribers`** wizard to send messages to all subscribers of all product plans that contain a specific API on your Developer Portal.
- **Emailing plan subscribers**
  As a site administrator, you can use the email **`Plan subscribers`** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal.
- **Emailing product subscribers**
  As a site administrator, you can use the email **`Product subscribers`** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal.
- **Token reference**
  You can use the email subscribers wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

Emailing all subscribers

As a site administrator, you can use the email **`All users`** wizard to email all the subscribers of all the consumer organizations on your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can use the **`All users`** wizard to send messages to all consumer organizations. You can select whether to email just the owner, or all of the members of the consumer organizations. Each recipient receives an individual email.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **`All users`**, and click Save.
4. Select the role of the recipients that you want to send the email to, then click Next.

- Owners - sends the email to the relevant consumer organization owners.
- Members - sends the email to all members of the relevant consumer organizations.

5. Enter the content of the email, subject, and body, and set the Text Format of the email:
   - Basic HTML
   - Plain text
6. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see Token reference.
7. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
   - **`Priority`**: Set the priority of the email. Many email programs ignore a priority setting.
   - **`Request receipt`**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
   - **`Additional headers`**: Set any additional headers to be sent with the email.
8. Use the remaining check boxes to configure what messages are to be sent and how:
   - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
   - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
   - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
9. Click Next.
10. Confirm that the email is ready to send, then click Next.
11. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.
    If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
12. Click Finish.

Results

You successfully sent an email to all members, or all of the owners of each consumer organization on your Developer Portal.

Emailing api subscribers

As a site administrator, you can use the email **`API subscribers`** wizard to send messages to all subscribers of all product plans that contain a specific API on your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can use the email **`API subscribers`** wizard to send messages to all subscribers of all product plans that contain a specific API in your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **`API subscribers`**, and click Save.
4. In the text box, enter the name of the API whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the role of the recipients that you want to send the email to, then click Next.
   - Owners - sends the email to the relevant consumer organization owners.
   - Members - sends the email to all members of the relevant consumer organizations.
6. Enter the content of the email, subject, and body, and set the Text Format of the email:
   - Basic HTML
   - Plain text
7. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see Token reference.
8. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
   - **`Priority`**: Set the priority of the email. Many email programs ignore a priority setting.
   - **`Request receipt`**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
   - **`Additional headers`**: Set any additional headers to be sent with the email.
9. Use the remaining check boxes to configure what messages are to be sent and how:
   - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
   - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.

- Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.

10. Click Next.
11. Confirm that the email is ready to send, then click Next.
12. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.
    If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
13. Click Finish.

Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to all product plans that contain the chosen API on your Developer Portal.

Emailing plan subscribers

As a site administrator, you can use the email **`Plan subscribers`** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can use the email **`Plan subscribers`** wizard to send messages to all subscribers of a chosen plan that belongs to a specific product on your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **`Plan subscribers`**, and click Save.
4. In the text box, enter the name of the product whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the name of the plan whose subscribers you want to email. Click Next.
6. Select the role of the recipients that you want to send the email to, then click Next.
   - Owners - sends the email to the relevant consumer organization owners.
   - Members - sends the email to all members of the relevant consumer organizations.
7. Enter the content of the email, subject, and body, and set the Text Format of the email:
   - Basic HTML
   - Plain text
8. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement.
   You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see Token reference.
9. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
   - **`Priority`**: Set the priority of the email. Many email programs ignore a priority setting.
   - **`Request receipt`**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
   - **`Additional headers`**: Set any additional headers to be sent with the email.
10. Use the remaining check boxes to configure what messages are to be sent and how:
    - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
    - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
    - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
11. Click Next.
12. Confirm that the email is ready to send, then click Next.
13. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.
    If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
14. Click Finish.

Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to the chosen plan within the chosen product, on your Developer Portal.

Emailing product subscribers

As a site administrator, you can use the email **`Product subscribers`** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

You can use the email **`Product subscribers`** wizard to send messages to all subscribers of any plan for a specific product in your Developer Portal community. For each consumer org, you can select whether to email just the owner or all members. Each recipient receives an individual email.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Navigate to People > Email subscribers.
3. Select **`Product subscribers`**, and click Save.
4. In the text box, enter the name of the product whose subscribers you want to email. If you start typing in the text box, you can choose from the options that you are given. Click Next.
5. Select the role of the recipients that you want to send the email to, then click Next.
   - Owners - sends the email to the relevant consumer organization owners.
   - Members - sends the email to all members of the relevant consumer organizations.
6. Enter the content of the email, subject, and body, and set the Text Format of the email:
   - Basic HTML
   - Plain text
7. Expand the REPLACEMENTS drop-down, and click Browse available tokens to view which tokens can be added to the subject and body fields for replacement. You can select a token to use by either typing it directly, or by clicking in the subject or body field. You are then given a list of available tokens, and you can insert the token that you want. For a reference list of tokens, see Token reference.
8. Expand the ADDITIONAL EMAIL OPTIONS drop-down to configure any additional settings:
   - **`Priority`**: Set the priority of the email. Many email programs ignore a priority setting.
   - **`Request receipt`**: Request a read receipt from your emails. Many email programs ignore these requests so it is not a definitive indication of how many people read your email.
   - **`Additional headers`**: Set any additional headers to be sent with the email.
9. Use the remaining check boxes to configure what messages are to be sent and how:
   - Send the message directly by using the Batch API. This option sends the emails directly by the Batch API. If not selected, then the emails are added to the spool and sent out by cron.
   - Send a copy of all messages to the sender. Because emails might contain context-specific tokens, a new email is generated for each recipient with the correct token replacement. This option sends a copy of all generated emails to the sender.
   - Send a copy of the original message to the sender. This option sends a single copy of the original email, subject, and body, without the replacement of any specified tokens.
10. Click Next.
11. Confirm that the email is ready to send, then click Next.
12. If you chose to send the emails by the Batch API, you see a progress page that indicates how many emails were sent. You are then redirected to a summary page that shows the total number of emails that were sent.
    If you chose to spool the delivery of the emails, you are redirected to the summary page that shows the total number of emails that are spooled.
13. Click Finish.

Results

You successfully sent an email to all members or all owners of each consumer org that contained a subscription to any plan within the chosen product on your Developer Portal.

Token reference

You can use the email subscribers wizard to specify context-specific tokens in the email subject and body. The tokens available and their replacement values depend on which flow is selected, for example, API, plan, product, or all.

Application Tokens

Tokens that are related to an individual Application.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Description	`[application:description]`	The description that is given to the application.	✓	✓	✓	
Enabled	`[application:enabled]`	The state that the application is in.	✓	✓	✓	
ID	`[application:id]`	The ID of the application.	✓	✓	✓	
Image URL	`[application:image_url]`	The URL of the image for the application.	✓	✓	✓	
Lifecycle State	`[application:lifecycle_state]`	The lifecycle state of the application.	✓	✓	✓	
Name	`[application:name]`	The name of the application.	✓	✓	✓	

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Title	`[application:title]`	The title of the application.	✓	✓	✓	

Consumer Organization Tokens

Tokens that are related to the individual consumer organization.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
ID	`[consumer-org:id]`	The ID of the consumer organization.	✓	✓	✓	✓
Name	`[consumer-org:name]`	The name of the consumer organization.	✓	✓	✓	✓
Title	`[consumer-org:title]`	The title of the consumer organization.	✓	✓	✓	✓

Product Tokens

Tokens that are related to an individual product.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
APIs	`[product:apis]`	A comma separated list of APIs (name:version) within the product.	✓	✓	✓	
ID	`[product:id]`	The ID of the product.	✓	✓	✓	
Image URL	`[product:image_url]`	The URL of the image for the product.	✓	✓	✓	
Name	`[product:name]`	The name of the product.	✓	✓	✓	
Plans	`[product:plans]`	A comma separated list of plan names within the product.	✓	✓	✓	
State	`[product:state]`	The state of the product.	✓	✓	✓	
Title	`[product:title]`	The title of the product.	✓	✓	✓	

Plan Tokens

Tokens that are related to an individual product plan.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Approval required	`[product-plan:approval]`	Whether the plan requires approval or not.		✓		
Description	`[product-plan:description]`	The description of the plan.		✓		
Name	`[product-plan:name]`	The name of a given plan within the product.		✓		
Rate limits	`[product-plan:rate-limits]`	A comma separated list of rate limit name value pairs that the plan contains.		✓		
Title	`[product-plan:title]`	The title of the plan.		✓		

API Tokens

Tokens that are related to an individual API.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
ID	`[api:id]`	The ID of the API.			✓	
Image URL	`[api:image_url]`	The URL of the image for the API.			✓	
Name	`[api:name]`	The name of the API.			✓	
OAI Version	`[api:oai_version]`	The OAI version of the API.			✓	
Protocol	`[api:protocol]`	The protocol of the API.			✓	
State	`[api:state]`	The current state the API is in.			✓	
Title	`[api:title]`	The title of the API.			✓	
Version	`[api:version]`	The version of the API.			✓	

User Tokens

Tokens that are related to the individual user account of the recipient.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Account Name	`[user:account-name]`	The login name of the user account.	✓	✓	✓	✓
Created	`[user:created]`	The date the user account was created.	✓	✓	✓>	✓
Display Name	`[user:display-name]`	The display name of the user account.	✓	✓	✓	✓
Email	`[user:mail]`	The email address of the user account.	✓	✓	✓	✓
First Name	`[user:first_name]`	The first name of the user.	✓	✓	✓	✓

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Last Name	`[user:last_name]`	The last name of the user.	✓	✓	✓	✓
Roles	`[user:roles]`	The user roles associated with the user account.	✓	✓	✓	✓
State	`[user:apic_state]`	List (text) field.	✓	✓	✓	✓
User ID	`[user:uid]`	The unique ID of the user account.	✓	✓	✓	✓

Current User Tokens

Tokens that are related to the current user who is initiating the sending of the emails.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Account name	`[current-user:account-name]`	The login name of the user account.	✓	✓	✓	✓
Display name	`[current-user:display-name]`	The display name of the user account.	✓	✓	✓	✓
Email	`[current-user:mail]`	The email address of the user account.	✓	✓	✓	✓

Site Information Tokens

Tokens that are related to site-wide settings and other global configuration.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Email	`[site:mail]`	The administrative email address for the site.	✓	✓	✓	✓
Login page	`[site:login-url]`	The URL of the site's login page.	✓	✓	✓	✓
Name	`[site:name]`	The name of the site.	✓	✓	✓	✓
Slogan	`[site:slogan]`	The slogan of the site.	✓	✓	✓	✓
URL	`[site:url]`	The URL of the site's front page.	✓	✓	✓	✓
URL (brief)	`[site:url-brief]`	The URL of the site's front page without the protocol.	✓	✓	✓	✓

Current Date Tokens

Tokens that are related to the current date and time.

Name	Token	Description	Product Subscribers	Plan Subscribers	API Subscribers	All Users
Custom format	`[current-date:custom:?]`	A date in a custom format. See the PHP documentation for details.	✓	✓	✓	✓
Fallback date format	`[current-date:fallback]`	A date in 'fallback' format. (Fri, 01/08/2021 - 14:19)	✓	✓	✓	✓
HTML Date	`[current-date:html_date]`	A date in 'html_date' format. (2021-01-08)	✓	✓	✓	✓
HTML Datetime	`[current-date:html_datetime]`	A date in 'html_datetime' format. (2021-01-08T14:19:19+0000)	✓	✓	✓	✓
HTML Month	`[current-date:html_month]`	A date in 'html_month' format. (2021-01)	✓	✓	✓	✓
HTML Time	`[current-date:html_time]`	A date in 'html_time' format. (14:19:19)	✓	✓	✓	✓
HTML Week	`[current-date:html_week]`	A date in 'html_week' format. (2021-W01)	✓	✓	✓	✓
HTML Year	`[current-date:html_year]`	A date in 'html_year' format. (2021)	✓	✓	✓	✓
HTML Yearless date	`[current-date:html_yearless_date]`	A date in 'html_yearless_date' format. (01-08)	✓	✓	✓	✓
Long format	`[current-date:long]`	A date in 'long' format. (Friday, January 8, 2021 - 14:19)	✓	✓	✓	✓
Medium format	`[current-date:medium]`	A date in 'medium' format. (Fri, 01/08/2021 - 14:19)	✓	✓	✓	✓
Raw timestamp	`[current-date:raw]`	A date in UNIX timestamp format (1610115559)	✓	✓	✓	✓
Short format	`[current-date:short]`	A date in 'short' format. (01/08/2021 - 14:19)	✓	✓	✓	✓
Time-since	`[current-date:since]`	A date in 'time-since' format. (6 minutes)	✓	✓	✓	✓

How to style the Developer Portal emails

You can apply a theme to the HTML emails that are sent by the Developer Portal by using CSS and templating.

Before you begin

You must have administrator access to complete this task.

You must have a custom theme. For information about how to create a custom theme, see Creating a sub-theme.

About this task

You can apply custom styling to the HTML content of the emails that are sent by the Developer Portal, such as the emails that are sent by the Email subscribers wizard. The custom styling consists of two parts: editing a twig template and, optionally, also editing a CSS file in your custom theme. See the following instructions for details. Note: These instructions apply only to emails that are sent by the Developer Portal. These instructions don't apply to the consumer invitation, activation, and password reset emails, as these emails are sent by the API Manager. For information about how to style API Manager emails, see Configuring notifications and Sending messages to consumer organization owners.

Procedure

1. Download the ibm-apic-mail-message.html.twig template from https://github.com/ibm-apiconnect/devportal/tree/master/ibm_apim/ibm_apic_mail/templates, and save it to your custom theme templates directory.
2. Edit the ibm-apic-mail-message.html.twig template to fit your custom theme.
   You can add styles directly into this twig template, or you can add them to the css/mail.css file in your custom theme, which will mean that they are automatically included in the HTML email content that is sent.

   If you are using an SCSS theme that has been generated by the Developer Portal theme generator, then you can put the custom SCSS mail styles into the mail-overrides.scss file. These styles will then be compiled into the mail.css file.

   Note: If you are using an older theme that was generated before API Connect Version 10.0.4.0, then the theme won't contain the mail theme support. However, you can workaround this by generating a new sub-theme with the same name, copying the overrides.css, or overrides.scss, as appropriate, logo.png, and *.theme files from the old theme into the new sub-theme, and then compress and install the new sub-theme as usual. (See Creating a sub-theme for details.)
3. Define a standard set of classes in the ibm-apic-mail-message.html.twig template that the Developer Portal administrators can use.
   The twig template lists the variables that can be used within it. The HTML **<body>** element is the actual content of the email message as defined within Drupal. For example, that content could be the information that's entered in the WYSIWYG editor of the Email subscribers wizard. Note that the HTML content in the WYSIWYG editor is more restricted for security reasons, for example, it's not possible to use inline styles in the HTML content in the editor. However, it is possible to use CSS class names, and then define the class in the mail.css file, which is then included in the emails. Therefore, it's recommended that you define a standard set of classes that the Developer Portal administrators can use when they're creating emails.

4. Optional: Add images to the ibm-apic-mail-message.html.twig template.
   Images can be used by adding a **** reference in the twig template. The **src** attribute for the image must be a fully qualified web URL, and must be externally accessible so that the email recipients can access the image. It's not possible to reference local images, they must be fully qualified URLs. It's also not possible to embed or attach images or other files in the emails.

   Note: The **logo** will be automatically set to the fully qualified URL for the site logo image.
5. Compress all of your sub-theme resources back into a .zip file, and install into the Developer Portal. See Installing additional themes for details.

What to do next

It is possible to have multiple email templates. For example for different modules, where one template could be used for the emails that are sent by the contact module, and another template for the emails that are sent by the Email subscribers wizard. This more advanced configuration can be done by using the module and key attributes, and is explained within the twig template itself. Essentially, you create a twig template that's called ibm-apic-mail-message.contact.html.twig, for example, to override the default template for only the contact module. Then the most specific template for a module is used by the Developer Portal, before falling back on the more general template.

Related tasks

- Creating a sub-theme
- Installing additional themes
- Applying a modified content type template

Working with roles in the Developer Portal

As a site administrator, you can create and customize site configuration roles in the Developer Portal.

Rather than assigning individual permissions to each user, permissions are assigned to roles, and then roles are assigned to users. Using roles gives administrators greater control over permissions, and makes it easy to assign and remove roles when necessary.

Administrators can assign the following site configuration roles to users in the Developer Portal:

Superuser
: The Superuser role has the highest level of permissions in the Developer Portal, and is the role that is assigned by default to the admin user, in addition to the Administrator role, when they enable a Developer Portal site in the API Manager. This role bypasses all content access control, which means that users that have a Superuser role can see all of the site content. This level of access can lead to some unexpected behavior when they access content such as their Consumer organizations and their applications. Careful consideration should be given before you assign users to a Superuser role.

Administrator

An Administrator can use the administrator dashboard to customize and configure the Developer Portal. An Administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps.

Content Author
: A Content Author is a curator of content for the Developer Portal. A Content Author can perform the following actions:

  - Set taxonomies on the content.
  - Provide custom icons for content.
  - Upload additional documentation files.
  - Create documentation pages in the Developer Portal.

Forum Moderator
: A Forum Moderator can moderate the content of a forum in the Developer Portal.

Note: Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.
To learn how to create and customize roles in the Developer Portal, use the following topic links:

- **Creating administrator users for the Developer Portal**
  You can create additional administrator users for an Developer Portal.
- **Assigning users to a role**
  Assign users to one or more roles in the Developer Portal to enable role specific permissions.
- **Assigning permissions to a role**
  You can assign permissions to a new role in the Developer Portal, and update the permissions for current roles.
- **Creating a new role**
  You can create a new role within the Developer Portal.

Creating administrator users for the Developer Portal

You can create additional administrator users for an Developer Portal.

Before you begin

You must have administrator access to complete this task.

About this task

After you enable a Developer Portal site in the API Manager, you are sent an email that contains a one-time log in link for the Developer Portal site Admin user. The Admin user can administer the Content Management System (CMS) capabilities of the Developer Portal. For more information about enabling a site, see Creating and configuring Catalogs.

However, you can also create additional administrative roles for users from within the Developer Portal by assigning the role to users that have access to the Developer Portal. A user with an Administrator role can use the administrator dashboard to customize and configure the Developer Portal. An Administrator is able to perform any task within the Developer Portal that does not involve the creation of APIs, Products, and Apps. You should be careful to ensure that only trusted users are given this access and level of control of your site.

Procedure

To create additional administrative users, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the List tab.
4. Select the check boxes for the target users.
5. From the drop-down list under the Action heading, select Add the Administrator role to the selected user(s).
6. Click Apply to selected items.

Results

You have assigned the selected users to the Administrator role.

Assigning users to a role

Assign users to one or more roles in the Developer Portal to enable role specific permissions.

Before you begin

You must have administrator access to complete this task.

Procedure

To assign users to a role, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the List tab.
4. Select the check boxes for the target users.
5. From the drop-down list under the Action heading, select the role that you want to assign.
6. Click Apply to selected items.

Results

You have assigned the selected users to the required role.

Assigning permissions to a role

You can assign permissions to a new role in the Developer Portal, and update the permissions for current roles.

Before you begin

You must have administrator access to complete this task.

Procedure

To assign or update permissions for a role, complete the following steps:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click People in the administrator dashboard.
3. Select the Permissions tab.
4. Assign a permission to a role, by selecting the check box in the relevant column for the permission that you want to assign.
5. Click Save permissions.

Results

You have assigned permissions to the target role.

Creating a new role

You can create a new role within the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To create a new role, complete the following steps:

1. Click People in the administrator dashboard.
2. Select the Roles tab.
3. Click Add role.
4. Enter a name for the new role in Role name, and click Save.
5. Your new role is displayed in the list of roles. You can drag any of the roles to re-order them. It's recommended to order roles from least permissive (for example, Anonymous user) to most permissive (for example, Administrator user). Users who aren't logged in have the Anonymous user role. Users who are logged in have the Authenticated user role, plus any other roles granted to their user account.

Results

You have created a new role in the Developer Portal.

What to do next

The new role is created with the same permissions as an Authenticated user role. You can click the Permissions tab and edit the permissions as required.

Forums

Create and control forums in the Developer Portal.

The Developer Portal lets you create threaded discussion boards, called forums, on your site. To learn how to configure forums, use the following topic links.

- **Creating a new forum**
  You can create a new forum in the Developer Portal.
- **Creating new forum containers**
  You can create new forum containers in the Developer Portal.
- **Configuring the creation of new forums for each API**
  You can choose whether or not to automatically create a new forum when an API is created in the Developer Portal.
- **Locking forum topics**
  You can lock forum topics in the Developer Portal.
- **Marking content in a forum as sticky**
  You can mark content in a forum as sticky in the Developer Portal.
- **Removing forum posts**
  You can remove forum posts in the Developer Portal.
- **Turning off forums in the Developer Portal**
  You can turn off forums in the Developer Portal by disabling the Forum module. This action prevents users from viewing, and contributing to, forums.

Creating a new forum

You can create a new forum in the Developer Portal.

Before you begin

You must have administrator or Forum moderator access to complete this task.

Procedure

To create a new forum, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Forums.
4. Click Add forum.
5. Fill in the Name and Description fields for the new forum.
6. From the Parent drop-down list, select the hierarchical position of the new forum.
7. Click Save. You have created a new forum.

Creating new forum containers

You can create new forum containers in the Developer Portal.

Before you begin

You must have administrator or Forum moderator access to complete this task.

Procedure

To create a new forum containers, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Structure in the administrator dashboard.
3. Click Forums.
4. Click Add container.
5. Fill in the Name and Description fields for the new forum container.
6. From the Parent drop-down list, select the hierarchical position of the new forum container.
7. Click Save. You have created a new forum container.

Configuring the creation of new forums for each API

You can choose whether or not to automatically create a new forum when an API is created in the Developer Portal.

Before you begin

You must have administrator access to complete this task.

Procedure

To configure the creation of new forums for each API, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Configuration in the administrator dashboard.
3. Under the SYSTEM heading, click IBM API Connect.
4. Select or clear the Automatically create a forum per API check box to turn the feature on or off.
5. Click Save configuration.
   You have configured the creation of new forums for each API.

Locking forum topics

You can lock forum topics in the Developer Portal.

Before you begin

You must have administrator or Forum moderator access to complete this task.

Procedure

To lock forum topics, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. Click Edit for the forum topic that you want to lock.
4. Click Comment settings. The view expands.
5. Select Closed .
6. Keeping the Published selected, click Save.
   You have disallowed any further replies and marked the topic as locked.

Marking content in a forum as sticky

You can mark content in a forum as sticky in the Developer Portal.

Before you begin

You must have administrator or Forum moderator access to complete this task.

Procedure

To mark content in a forum as sticky, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. From the Action drop-down menu, select Make content sticky.
4. Select the check boxes next to the forum topics that you want to make sticky.
5. Click Apply to selected items.
   You made the selected forum content sticky.

Removing forum posts

You can remove forum posts in the Developer Portal.

Before you begin

You must have administrator or Forum moderator access to complete this task.

Procedure

To remove forum posts, complete the following tasks:

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Click Content in the administrator dashboard.
3. From the Action drop-down menu, select Delete content.
4. Select the check boxes next to the forum posts that you want to remove.
5. Click Apply to selected items.
   You removed the selected forum posts.

Turning off forums in the Developer Portal

You can turn off forums in the Developer Portal by disabling the Forum module. This action prevents users from viewing, and contributing to, forums.

Before you begin

You must have administrator access to complete this task.

About this task

To turn off forums, you must first remove all Forums terms, and then you can disable the Forum module.

Procedure

To turn off forums, complete the following steps.

1. If the administrator dashboard is not displayed, click Manage to display it.
2. Delete all Forums terms by completing the following steps:
    a. Click Structure > Taxonomy > Forums in the administrator dashboard.
    The list of Forums terms is displayed.
    b. For each item in the list of terms, select Delete from the drop-down list in the OPERATIONS column, and then click Delete again to confirm.
    All of the Forums terms are deleted.
3. Disable the Forum module by completing the following steps:
    a. Click Extend > Disable module in the administrator dashboard.
    b. Select the Forum module, then click Disable.
    c. The display then shows a list of forum configuration that will also be removed if the Forum module is disabled. Click Disable again to confirm that you agree to this removal.

Results

You successfully turned off forums in the Developer Portal.
If you want, you can enable the Forum module again by clicking Extend, selecting the Forum module on the List tab, and clicking Enable.

Reports

You can view reports about the general configuration and status of your Developer Portal site.

Use the Reports section in the Developer Portal to view information about the following areas:

Available updates
: Status report about available updates for your installed modules and themes.

Node Type Count
: Status report about how much of each content type there is in your portal site, and whether it is published or not. It can also provide totals of how many users there are of each role.

Available translation updates
: Status report about available interface translations for your installed modules and themes.

Field list
: Overview of fields on all entity types.

Status report
: Short overview of your site's parameters, as well as any problems detected with your installation.

Views plugins
: Overview of the plugins that are used in all views.

Extend

You can extend the Developer Portal by creating and installing custom modules, or installing additional Drupal contributed modules.

Modules provide functionality to your site, and the *core modules* that are included in the standard Developer Portal installation, provide all of the basic functions that most sites need. However, some of the core modules are not enabled by default, so you might want to examine the module list under the Extend menu of your site, to see whether the functionality you require is available from a core module before starting to investigate custom and contributed modules. If you still require additional custom or contributed modules, consider the following topics to help you create and implement modules consistently.
Note: The Developer Portal contains a Rules framework for event based actions. Some of those actions and triggers might exhibit unexpected behavior. A workaround is to create a custom module instead, because the same events and actions are possible there.

- **Custom module development: an introduction to Drupal tools for PHP development**
  Some guidance about the Drupal tools and environment considerations when developing in PHP, to help when writing custom modules to extend your Developer Portal functionality.
- **Custom module development: background and prerequisites**
  You can create custom modules in the Developer Portal to extend functionality. The following information provides a getting started developer guide to custom module development.

- **Custom module development: creating a module skeleton**
  You can extend your Developer Portal site by creating custom modules. The starting point for a new module is to create a module skeleton.
- **Custom module development: using hooks**
  You can use hooks in custom modules to alter the behavior of Drupal core or other modules in your Developer Portal.
- **Installing custom modules**
  You can extend your Developer Portal site by installing custom modules that you created, and also installing contributed modules from the Drupal community.
- **Deleting custom modules**
  You can delete custom modules from your Developer Portal if you want to remove them entirely from your site.
- **Disabling modules**
  You can disable an entire module in the Developer Portal if you want to improve performance, or remove functionality.
- **Using Developer Portal metadata on API Manager**
  For content created or updated after Version 10.0.1.0, the Developer Portal propagates more metadata to API Manager for `consumerorgs`, `users`, and `applications`.
- **Using a custom module for payment method creation**
  You can offer free use of your products to your customers or you can configure your Developer Portal to bill your customers when they use your products. You can use a custom module to configure creation of payment methods.

Related information

- Extending Drupal

Custom module development: an introduction to Drupal tools for PHP development

Some guidance about the Drupal tools and environment considerations when developing in PHP, to help when writing custom modules to extend your Developer Portal functionality.

The following sections provide an overview of the key automation tools and environment considerations for PHP development in the Developer Portal. For a complete list of Drupal tools, see Development tools overview on Drupal.org, however, note that you are not permitted to access the shell inside the Developer Portal containers. Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

Devel, Devel Generate Kint, and Webprofiler development modules

The Devel, Devel Generate Kint, and Webprofiler modules are all packaged within the Devel module. These modules provide the following development support:

- *Devel* - Helper functions for Drupal developers.
- *Devel Generate Kint* - Accelerate development of your site or module by quickly generating nodes, comments, terms, users, and more.
- *Kint* - Pretty print of variables with `kint($my_var)`.
- *Webprofiler* - Port of the Symfony WebProfiler Bundle as a Drupal module.

For more information, see Devel module on drupal.org, and Devel on drupalship.org.

Admin Toolbar module

The Admin Toolbar module improves the default Drupal Toolbar by transforming it into a drop-down menu, providing fast access to the administration pages. It also adds extra links to the Drupal icon on the admin toolbar to development tools such as flush all caches, run cron, and run updates.

For more information, see Admin toolbar module on drupal.org.

Examples for developers project

The Examples for Developers project aims to provide high-quality, well-documented API examples for a broad range of Drupal core functionality.

The project contains many modules that illustrate best practices for implementing Drupal core APIs. For example there are examples for Block, Cache, Config and Content Entity, Cron, Database API, Email, Events, Form API, Field, Field Permission, File, Hooks, Javascript, Node Type, Page, Pager, PHPUnit, Plugin Type, Queue, Simple Test, Stream Wrapper, Table Sort, Testing, and Tour.

For more information, see Examples for developers project on drupal.org.

Coding standards

The Drupal Coding Standards apply to code within Drupal and its contributed modules. For more information, see Coding standards on drupal.org.

Related tasks

- Installing custom modules

Related information

- Drupal tools for developers

Custom module development: background and prerequisites

You can create custom modules in the Developer Portal to extend functionality. The following information provides a getting started developer guide to custom module development.

Custom modules are written in PHP, which is an open source server-side scripting language that is used for creating dynamic web pages. For general information on the Drupal tools for PHP development, see Custom module development: an introduction to Drupal tools for PHP development.

The following sections provide an overview of custom module development, with links to learn more about the required development languages and concepts. Creating custom modules affects the source of your Developer Portal site, so it is recommended that you gain Drupal experience and PHP knowledge before you start creating modules.

Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

Object-Oriented Programming (OOP)

OOP is established as best practice for development, so ensure your knowledge of OOP is good. The PHP documentation on Classes and Objects is a good place to start. For a general overview of PHP best practices, see PHP The Right Way. The following resources are also helpful:

- Object-oriented programming (on Wikipedia)
- Object Oriented Programming with PHP (on phpro.org)
- Object-Oriented PHP for Beginners (on tutsplus.com)
- Object Oriented Programming in PHP (on tutorialspoint.com)

Drupal also uses some common design patterns, so you need to ensure you have a basic understanding of these. See the following pattern resources:

- The Factory Pattern (on phptherightway.com), and Late Static Bindings (on php.net)
- Software design pattern (on Wikipedia)
- Programming Foundations: Design Patterns (on lynda.com)

PHP namespaces

You must be familiar with the concept of namespaces in PHP. In most cases, the Drupal code is namespaced based on the module that code belongs to.

For example, the namespace for **`block.module`** is:

```
namespace Drupal\block;
```

When you create Drupal modules, it is important to consider that PHP has a global namespace, and that function names must be unique. You are recommended to prefix the name of any methods with the module name.

For example, if you have a module that is named **`custom_module`**, a **`create`** method within it is called **`custom_module_create()`**.

For more information about namespaces, see the following resources:

- Drupal namespace standards (on drupal.org)
- How to use PHP namespaces (on sitepoint.com)
- PHP Namespaces (on php.net)

Dependency injection

It is important that you have a baseline understanding of dependency injection. Drupal makes heavy use of this concept, so you will need this understanding to be able to access and make use of many of the core APIs.

To find out more about dependency injection, see Dependency Injection (on phptherightway.com), as well as the additional articles that are linked to on that page. See also Services and dependency injection in Drupal (on drupal.org).

Symfony

Symfony is a set of reusable PHP components and a PHP framework for web projects. Drupal borrows from this framework in order to reduce code duplication across various PHP projects. Much of the code that Drupal uses to handle routing, sessions, and the services container, amongst other things, is borrowed from Symfony.

To understand how Symfony works, see the Symfony Documentation on symfony.com.

Other useful resources

- Drupal API Annotations - list of the different annotation types that Drupal uses for plugin discovery, and to provide additional context/meta-data for the code that's being executed.
- Plugin API overview - overview about how plugins are used in Drupal.
- Tools are available for checking your custom modules, such as drupal_check on GitHub, which checks Drupal code for deprecations.

Related tasks

- Installing custom modules

Related information

- Creating custom modules

Custom module development: creating a module skeleton

You can extend your Developer Portal site by creating custom modules. The starting point for a new module is to create a module skeleton.

Before you begin

You must have administrator access to complete this task.

You must also have experience in Drupal and PHP development. For more information, see Custom module development: an introduction to Drupal tools for PHP development and Custom module development: background and prerequisites.

Important:

- You are not permitted to include any IBM® API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

About this task

Every custom module creation starts with creating an .info.yml file. This file, or module skeleton, is used to store the metadata about the custom theme. After the module skeleton is created, the module can be installed onto your site by using the Extend menu on the administrator dashboard of the Developer Portal UI. However, without any module specific code, your site won't be affected.
The following instructions guide you through creating a custom module skeleton. What to do next in your custom module creation depends on what you want your module to do. Options are shown at the end of this task.

Procedure

1. Choose a short name, also known as a machine name, for your custom module.
   The machine name is used in several files and functions in your module, as well as by Drupal core to programmatically refer to your module. The name must conform to the following rules:
   - It must start with a letter.
   - It must contain only lowercase letters and underscores. (Note that if uppercase letters are used in the name, Drupal won't recognize any hook implementations in your module.)
   - It must not contain any spaces.
   - It must be unique within your Developer Portal site.
   - It should not be any of the following reserved terms: **`src`**, **`lib`**, **`vendor`**, **`assets`**, **`css`**, **`files`**, **`images`**, **`js`**, **`misc`**, **`templates`**, **`includes`**, **`fixtures`**, **`Drupal`**.
2. Create a module folder in a location of your choice, for example, /modules/*module_name*.
   You don't have to name the folder the same name as your module machine name, however, you must remember to use the machine name programmatically within your module's code and filenames.
3. Create a *module_name*.info.yml file in your module folder.
   Where *module_name* is the short name, or machine name, that you chose in Step 1.
4. Insert the metadata for your custom module into the *module_name*.info.yml file as follows:

```
name: Example module name
description: Description of the module.
package: Custom

type: module
core: 8.x
core_version_requirement: '^8 || ^9'
version: 1.0

dependencies:
  - module_name

test_dependencies:
  - module_name

configure: example.settings

php: 7.1

hidden: true
```

   where
   - **`name`** (required) is the name of your module, and is the text that is shown on the module administration page in the Developer Portal UI.
   - **`description`** (optional) is a description of your module, and is the text that is shown on the module administration page in the Developer Portal UI.

- **`package`** (optional) is a key that enables you to group similar modules together.
- **`type`** (required) is the type of extension that you are creating, for example, a module, theme or profile. In this instance you are creating a module.
- **`core`** (optional) specifies with which Drupal core version your module is compatible.
- **`core_version_requirement`** (required) specifies which versions of Drupal your module is compatible with. This example specifies that the module is compatible with all versions of Drupal 8 and 9.
- **`version`** (required) is the version number of your module. Note that **`version`** is not required if you are publishing your module on Drupal.org.
- **`dependencies`** (optional) is a YAML list of any modules that your module depends on. They must be namespaced in the format ***`module_name`***, where *module_name* is the module's machine name.
- **`test_dependencies`** (optional) is list of any modules (in the same format as **`dependencies`**) that are needed to run certain automated tests for your module on Drupal's automated test runner (DrupalCI), but are not needed as module dependencies in general.
- **`configure`** (optional), if your module offers a configuration form, specify the route to this form here.
- **`php`** (optional) is the minimum PHP version that is required for your module. For the Developer Portal the minimum PHP version is 7.1.
- **`hidden`** (optional) is an option that, if set, will hide your module from the Extend section of the administrator dashboard on the Developer Portal UI. This option might be useful if your module only contains tests, or if the module is meant as an example to other developers.

5. Save your *module_name*.info.yml.

Results

You successfully created your module skeleton.

Example

An example .info.yml file:

```
name: User field example
type: module
description: Developer Portal tutorial about a user field example
package: IBM API Connect Tutorials
core: 8.x
core_version_requirement: '^8 || ^9'
version: 8.x-0.0.1
dependencies:
  - ibm_apim
  - auth_apic
```

What to do next

You now need to create the appropriate content for your custom module, depending on what you want the module to do. For information about the various options, see Creating custom modules on Drupal.org. There is also a walkthrough example of creating a basic custom module that is available on Drupal.org, see: A "Hello World" custom page module.
Note: If you are defining URL paths in the routing file of your module, you must consider any potential implications of those paths, and avoid conflicts with other parts of the Developer Portal. For example, you are recommended not to prefix any paths with **`ibm_apic`**.
After you have created your custom module content, you need to package your custom module files up into a zip, tar, tgz, gz, or bz2 file. This file can then be installed onto your Developer Portal site by using the Extend menu on the administrator dashboard. For more information, see Installing custom modules.

Related information

- Let Drupal know about your module with an .info.yml file

Custom module development: using hooks

You can use hooks in custom modules to alter the behavior of Drupal core or other modules in your Developer Portal.

Custom modules enable you to extend the functionality of your Developer Portal site. For more information about how to create custom modules, see Custom module development: an introduction to Drupal tools for PHP development, and Custom module development: background and prerequisites.

Hooks are one of the ways that custom modules can use to interact with other modules and with Drupal core subsystems. The following sections provide an overview of hooks, and a list of the IBM® API Connect specific hooks.

Important:

- You are not permitted to include any IBM API Connect modules within any custom modules that you create. Also, directly editing any API Connect themes, modules, included modules, or Drupal core on the file system is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.
- All custom development is your responsibility. Although the use of custom modules and themes is supported, IBM API Connect do not provide support in their development or modification.

About hooks

Hooks define functions that alter the behavior of Drupal core. So one way for custom modules to alter these functions, is to use hooks. Hooks are specially-named functions that a module defines (this is known as *implementing the hook*), these hooks are then discovered and called at specific times to alter or add to the base behavior or data (this is known as *invoking the hook*). Each hook has a name (for example: **`hook_batch_alter()`**), a defined set of parameters, and a defined return value. Your custom modules can implement hooks that are defined by Drupal core, by API Connect, or by other modules that they interact with. Your custom modules can also define their own hooks in order to let other modules interact with them. For more information, see Understanding Hooks on Drupal.org.

Note: If you write a hook implementation function, the function must not throw errors or exceptions, or use **`alert`** level log messages. Errors, exceptions, and **`alert`** level log messages can break all the processing in the parent code. For example, if you write a hook that calls an external server when an application is updated, the code needs

to have error handling in place to handle the server not being found, or not returning the expected result. Otherwise, Drupal stops parsing the function and the application might not get updated correctly. It can cause webhook and snapshot parsing to abort leaving the portal in an inconsistent state.
For a list of all the hooks that are available on Drupal core, and how to implement them in your custom module, see Drupal API Hooks.

For a list of all the API Connect specific hooks, see the following section.

Using hooks in custom modules

To invoke a hook you need to add a function to your .module file in your custom module, and prefix the hook with your custom module name. For example, when using an API Connect hook in a custom module, you must replace the word **hook** in the hook name with the name of your custom module. For example:

```
hook_apic_app_create
```

should be referenced in your custom module as:

```
moduleName_apic_app_create
```

See also the following example .module file that shows a custom module called user_field_example:

```
/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
  if ($form_id === 'user_register_form') {
    // add our validator to the #validate array for the user_register_form
    $form['#validate'][] = 'user_field_example_validate_department_code';
  }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 *   where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
  if (isset($form_state->getValue('field_department_code')[0]['value'])) {
    $dept_code = $form_state->getValue('field_department_code')[0]['value'];
    $valid = preg_match('/^DEP\d{3}$/', $dept_code);
    if (!$valid) {
      // if the value is not valid then set an inline error on the relevant field
      $form_state->setErrorByName('field_department_code', t('Invalid department code.'));
    }
  }
}
```

API Connect specific hooks

The following tables list the hooks that are specific to the API Connect Developer Portal. Each section contains a link to the .php file on the API Connect Developer Portal repository on GitHub for those hooks. This file contains examples of how you can use the hooks in your modules.

Hooks about Applications

The following table lists the hooks that relate to Applications in the Developer Portal. For examples of how to use Application hooks, see apic_app/apic_app.api.php on GitHub.

Table 1. Hooks about Applications

Hook name	Description
`hook_apic_app_create`	Triggered when an Application is created.
`hook_apic_app_update`	Triggered when an Application is updated.
`hook_apic_app_pre_delete`	Triggered when an Application is deleted, before the node deletion or cascade has happened.
`hook_apic_app_post_delete`	Triggered when an Application is deleted, after the node deletion or cascade has happened.
`hook_apic_app_promote`	Triggered when an Application is promoted.
`hook_apic_app_creds_create`	Triggered when a new set of credentials is created for an Application.
`hook_apic_app_creds_update`	Triggered when a set of credentials is updated for an Application.
`hook_apic_app_creds_delete`	Triggered when a set of credentials is deleted for an Application.
`hook_apic_app_subscribe`	Triggered when a subscription is created.
`hook_apic_app_migrate`	Triggered when a subscription is migrated to a new Plan.
`hook_apic_app_unsubscribe`	Triggered when an Application is unsubscribed from a Plan.
`hook_apic_app_image_create`	Triggered when a custom Application image is created.
`hook_apic_app_image_delete`	Triggered when a custom Application image is deleted.
`hook_apic_app_clientid_reset`	Triggered when a credential client ID is reset.
`hook_apic_app_clientsecret_reset`	Triggered when a credential client secret is reset.
`hook_apic_app_modify_getplaceholderimage_alter`	Alter the Application placeholder image provided in \Drupal\apic_app\Application::getPlaceholderImage(). Can be used to define a specific placeholder image to use when the consumer has not uploaded their own custom image for their Application.
`hook_apic_app_modify_getimageforapp_alter`	Alter the Application placeholder image provided in \Drupal\apic_app\Application::getImageForApp(). Can be used to provide a full path to a specific image to use for an Application that overrides any custom image that might have been uploaded.

Hook name	Description
`hook_apic_app_modify_client_id_reset_alter`	Alter the client ID provided by API Manager when the ID is reset.
`hook_apic_app_modify_client_secret_reset_alter`	Alter the client secret provided by API Manager when the secret is reset.
`hook_apic_app_modify_create_alter`	Alter the credentials provided by API Manager when a new Application is created.
`hook_apic_app_modify_credentials_create_alter`	Alter the credentials provided by API Manager when new credentials are created.

Hooks about Consumer organizations

The following table lists the hooks that relate to Consumer organizations in the Developer Portal. For examples of how to use Consumer organization hooks, see consumerorg/consumerorg.api.php on GitHub.

Table 2. Hooks about Consumer organizations

Hook name	Description
`hook_consumerorg_create`	Triggered when a Consumer organization is created.
`hook_consumerorg_update`	Triggered when a Consumer organization is updated.
`hook_consumerorg_pre_delete`	Triggered when a Consumer organization is deleted, before the node deletion or cascade has happened.
`hook_consumerorg_post_delete`	Triggered when a Consumer organization is deleted, after the node deletion or cascade has happened.
`hook_consumerorg_payment_method_create_alter`	Triggered to allow modification of the payment method creation form.
`hook_consumerorg_myorg_tabs_alter`	Triggered to allow more tabs to be added to the `my organization` page.

Hooks about APIs

The following table lists the hooks that relate to APIs in the Developer Portal. For examples of how to use API hooks, see apic_api/apic_api.api.php on GitHub.

Table 3. Hooks about APIs

Hook name	Description
`hook_apic_api_create`	Triggered when an API is created.
`hook_apic_api_update`	Triggered when an API is updated.
`hook_apic_api_delete`	Triggered when an API is deleted.

Hooks about Products

The following table lists the hooks that relate to Products in the Developer Portal. For examples of how to use Product hooks, see product/product.api.php on GitHub.

Table 4. Hooks about Products

Hook name	Description
`hook_product_create`	Triggered when a Product is created.
`hook_product_update`	Triggered when a Product is updated.
`hook_product_delete`	Triggered when a Product is deleted.

Other IBM API Connect Hooks

The following table lists the hooks that can be for other uses.

Table 5. Other APIC Hooks

Hook name	Description
`hook_ibm_apim_subscription_wizard_summary_alter`	Triggered to allow modification of the summary page of the subscription wizard.

Related tasks

- Installing custom modules

Related information

- Creating custom modules

Installing custom modules

You can extend your Developer Portal site by installing custom modules that you created, and also installing contributed modules from the Drupal community.

Before you begin

You must have administrator access to complete this task.

About this task

You can add extra functions to your site by installing new custom or contributed modules. For more information on creating custom modules, see Custom module development: background and prerequisites. Contributed modules are modules that are contributed by Drupal community members. To search for contributed modules, go to Drupal Download & Extend. You can install both types of modules into your Developer Portal site by using the following instructions.
Note: The following modules are unsupported and their installation is blocked in the Developer Portal:

- `advagg`
- `advagg_bundler`
- `advagg_css_minify`
- `advagg_ext_minify`
- `advagg_js_minify`

- `advagg_mod`
- `advagg_validator`
- `backup_migrate`
- `content_sync`
- `delete_all`
- `devel_themer`
- `domain`
- `php`
- `simple_html_mail`
- `statistics`
- `swiftmailer`
- `tfa`
- `theme_editor`

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The modules that are displayed with a selected checkbox are enabled, the modules that are displayed without a selected checkbox are not enabled.
3. Click + Install new module, and either complete the Install from a URL field, or click Choose file to upload a *filename*.tar.gz module file from your local computer.
4. Click Install.
   The Update manager confirms that the module was installed successfully.
5. Click Enable newly added modules to return to the List tab for the Extend page.
6. Find your newly added module in the list of modules, select its checkbox, and click Enable.
   Your newly added module is now enabled and available for you to use in the Developer Portal.

Deleting custom modules

You can delete custom modules from your Developer Portal if you want to remove them entirely from your site.

Before you begin

You must have administrator access to complete this task.

Note that any custom modules that you want to delete must be disabled first, including disabling any associated sub-modules. For more information, see Disabling modules. Any modules that are shipped with IBM® API Connect cannot be deleted.

Procedure

1. Log in to the Developer Portal as an administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Extend.
4. Click the Delete tab.
   The list of disabled custom modules that can be deleted is displayed. Note that any disabled modules that have sub-modules that are still enabled, are prevented from being disabled.
5. After you have found the module that you want to disable, select the check box next to the module, then click Delete.
6. At the confirmation dialogue, select Delete to delete the module completely from your site, or select Cancel to cancel the operation.

Results

You successfully deleted the module.

What to do next

To install a module, see Installing custom modules.

Disabling modules

You can disable an entire module in the Developer Portal if you want to improve performance, or remove functionality.

Before you begin

You must have administrator access to complete this task.

Note that the disable process removes all data related to a module.

Procedure

1. Log in to the Developer Portal as an administrator.

2. If the administrator dashboard is not displayed, click Manage to display it.
3. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed, some of which are enabled (displayed with a selected check box), and some of which are not enabled (where the check box is not selected).
4. Click the Disable tab.
   The list of modules that can be disabled is displayed. Note that any modules that are required by other enabled modules, are prevented from being disabled.
5. Either enter the name of the module that you want to disable in the Filter by name or description field, or scroll through the list of modules.
6. After you have identified the module that you want to disable, select the check box next to the module, then click Disable.
7. At the confirmation dialogue, select Disable to disable the module and remove any data that relates to it, or select Cancel to cancel the operation.

Results

You successfully disabled the module.

What to do next

To enable a module, return to the List tab, select the check box next to the module that you want to enable, and click Enable.
You can also delete custom modules completely from your Developer Portal site; see Deleting custom modules.

Using Developer Portal metadata on API Manager

For content created or updated after Version 10.0.1.0, the Developer Portal propagates more metadata to API Manager for **consumerorgs**, **users**, and **applications**.

This metadata might be values that the user provided by adding extra fields to content types, or added programmatically by using custom modules.

The benefit of this is that when you add custom fields to **consumerorgs**, **users**, and **applications**, this data is now also stored as metadata in API Manager. Storing this data allows the Developer Portal to correct or repopulate the custom fields from a webhook or a snapshot.

Related tasks

- Installing custom modules

Related information

- Creating custom modules

Using a custom module for payment method creation

You can offer free use of your products to your customers or you can configure your Developer Portal to bill your customers when they use your products. You can use a custom module to configure creation of payment methods.

Before you begin

You must have administrator access to complete this task.

You must have configured billing support in API Manager.

Create your custom module. Your custom module needs to implement the `consumerorg_payment_method_create_alter` hook, so it needs to have a function like this:

```
my_custom_module_consumerorg_payment_method_create_alter(&$form, $integration, $billing) {
// do this payment function with $form
}
```

For more information, see Custom module development: background and prerequisites.

Procedure

1. If the administrator dashboard is not displayed, click Manage to display it.
2. In the administrator dashboard, click Extend.
   The List tab for the Extend page opens, and the list of installed modules is displayed. The list shows all the modules that are installed. The enabled modules are displayed with a selected checkbox. The disabled modules are displayed without a selected checkbox.
3. Click + Install new module, and either complete the Install from a URL field, or click Choose file to upload a *filename*.tar.gz module file from your local computer.
4. Click Install.
   The Update manager confirms that the module was installed successfully.
5. Click Enable newly added modules to return to the List tab for the Extend page.
6. Find your newly added module in the list of modules, select its checkbox, and click Enable.
   Your newly added module is now enabled and available for you to use in the Developer Portal.
7. In the administrator dashboard, click Configuration > System > IBM API Connect Billing Settings in the administrator dashboard.
8. Change the drop-down option to be your newly enabled module for your billing provider.

Results

You successfully enabled your payment method by using a custom module.

Managing the Developer Portal by using the toolkit CLI

You can perform operations on your Developer Portal by using the API Connect toolkit CLI.

- **Getting started with the Portal CLI commands**
  The toolkit CLI provides developer portal-specific commands for managing your portal servers and sites.
- **Developer Portal CLI commands**
  A summary of the commands that are supported in the Developer Portal CLI.
- **Scenarios that use the Portal CLI**
  Here are some scenarios of how you can use the Portal commands in the toolkit CLI to accomplish tasks that are more complex to complete, or not possible in the Developer Portal UI.

Getting started with the Portal CLI commands

The toolkit CLI provides developer portal-specific commands for managing your portal servers and sites.

Procedure

1. Install the toolkit: Setting up the API Connect toolkit
2. Log in to the Management server. Toolkit CLI Login.

   ```
   apic login --server <management hostname/api base url> --username <username> --realm <realm>
   ```

   Where:
   - *<management hostname/api base url>* is the FQDN or base URL of your management API endpoints, for example: `mgmt.example.com` or `https://mgmt.example.com`.
   - *<username>* is either the admin user or provider org owner, depending on whether you want to run cloud or provider organization scoped operations.
   - *<realm>* is the user registry realm for the specified *<username>*. By default this is `admin/default-idp-1` for the admin user, and `provider/default-idp-2` for provider organization users.

   Note: All portal CLI commands are either cloud admin or provider org scoped.
3. View the available toolkit portal commands by using `-m portaladmin -h`. Example:

   ```
   apic -m portaladmin -h
   ...Usage:
     apic [flags]
     apic [command]

   Available Flags:
         --accept-license        Accept the license for API Connect
         --debug                 Enable debug output
         --debug-output string   Write debug output to file
     -h, --help                  Help for apic
         --live-help             Enable or disable tracking of limited usage information
     -m, --mode string           Toolkit operation mode (default "apim")

   Available Commands:

     Creating applications and artifacts
        config                               Manage configuration variables

     Publishing to the cloud
        api                                  Api operations
        apic-config                          Apic Config operations
        application                          Application operations
        backups                              Backups operations
        consumer-org                         Consumer Org operations
        custom-module                        Custom Module operations
        custom-theme                         Custom Theme operations
        custom-translation                   Custom Translation operations
        custom-webserver-page                Custom Webserver Page operations
        drupal-config                        Drupal Config operations
        drupal-state                         Drupal State operations
        entity                               Entity operations
        factory-reset                        Factory Reset operations
        ip-security-enabled                  Ip Security Enabled operations
        modules                              Modules operations
        php-memory                           Php Memory operations
        platforms                            Platforms operations
        product                              Product operations
        security                             Security operations
        service-ip-allowlist                 Service Ip Allowlist operations
        site                                 Site operations
        site-config                          Site Config operations
        sites                                Sites operations
        themes                               Themes operations
        twig                                 Twig operations

     Other commands
   ```

```
    licenses                              Review the license for API Connect
    validate                              Validate an API or product definition
    version                               Get the APIConnect toolkit version


Additional help topics:

Use "apic [command] --help" for more information about a command.
```

4. To view command-specific help, run the command with `-h`:

```
apic -m portaladmin php-memory -h
Usage: apic php-memory:list [flags]

Php Memory operations

Flags:
      --format string                 Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                          Help for list
      --portal_service_name string    The name of the portal service (required)
  -s, --server string                 management server endpoint (required)

Type "apic [command] --help" for help on the following related commands:
      php-memory:list    List the PHP memory limit on the platform.
      php-memory:update  Set the PHP memory limit on the portal platform.

apic -m portaladmin php-memory:list -h
Usage: apic php-memory:list [flags]

List the PHP memory limit on the platform.

Lists the PHP memory_limit setting on the portal platform.

Flags:
      --format string                 Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
      --portal_service_name string    The name of the portal service (required)
  -s, --server string                 management server endpoint (required)
```

Developer Portal CLI commands

A summary of the commands that are supported in the Developer Portal CLI.

Authenticating

Use the **`apic login`** command to authenticate, and the **`apic logout`** command to remove your local authentication credentials. For more information, see Getting started with the Portal CLI commands.

Note: When you authenticate successfully, your credentials are stored, in plain text, in the file Linux .netrc or Windows _netrc. Therefore, you must set the file permissions in such a way that your credentials are not accessible by others.

Command summary

The following tables summarize **`apic --mode portaladmin`** commands.

Table 1. Summary of the `portaladmin` commands

Command	Description	Subcommands
`apic api`	Manage your **`api`**.	• **`apic api:add-attachment`** - Add an attachment for an API • **`apic api:add-tag`** - Add a tag for an API. • **`apic api:get`** - Get an API from a portal service. • **`apic api:get-document`** - Get a specific entire API document from the portal of the given organization and catalog. • **`apic api:list`** - List the APIs of the given organization and catalog. • **`apic api:set-icon`** - Set an icon for an API. For more information, see Using the api commands.
`apic apic-config`	Manage your **`apic-config`**.	• **`apic apic-config:get`** - Get the configuration of a site. For more information, see Using the apic-config commands.
`apic application`	Manage your **`application`**.	• **`apic application:get`** - Get an application from a portal service. • **`apic application:list`** - List the applications on a portal service. For more information, see Using the application commands.
`apic backups`	Manage your **`backups`**.	• **`apic backups:list`** - List the backups of a portal service. For more information, see Using the backups commands.
`apic consumer-org`	Manage your **`apic consumer-org`**.	• **`apic consumer-org:get`** - Get a consumer org from a portal service. • **`apic consumer-org:list`** - List the consumer orgs on a portal service. For more information, see Using the consumer-org commands.

Command	Description	Subcommands
`apic content`	Manage your `content` entity types.	• `content:create-export` - Create a task to export a .tgz file containing your site content entities. • `content:create-import` - Create a task to import an archive of your site content. • `content:delete-export` - Cancel any currently running `content:create-export` tasks, and delete any related artifacts. • `content:delete-import` - Cancel any currently running `content:create-import` tasks, and delete any related artifacts. • `content:get-export-status` - Return the status of a specific export task. • `content:get-export` - Stream the content of a specific export task to a .tgz file. • `content:get-import-status` - Return the status of a specific import task. • `apic content:list` - List all of the entities in your site for a given content entity type and, if required, bundle. • `apic content:list-type` - List the content entity types that are exportable from your site. For more information, see Using the content commands.
`apic custom-module`	Manage your `custom modules`.	• `apic custom-module:create-export` - Create an export of a `custom module`. • `apic custom-module:create-import` - Create an import of a `custom module`. • `apic custom-module:delete-export` - Delete an export of a `custom module`. • `apic custom-module:delete-import` - Delete an import of a `custom module`. • `apic custom-module:get-export` - Get an export of a `custom module`. • `apic custom-module:get-export-status` - Get the status of an export of a `custom module`. • `apic custom-module:get-import-status` - Get the status of an import of a `custom module`. For more information, see Using the custom-module commands.
`apic custom-theme`	Manage your `custom themes`.	• `apic custom-theme:create-export` - Create an export of a `custom theme`. • `apic custom-theme:create-import` - Create an import of a `custom theme`. • `apic custom-theme:delete-export` - Delete an export of a `custom theme`. • `apic custom-theme:delete-import` - Delete an import of a `custom theme`. • `apic custom-theme:get-export` - Get an export of a `custom theme`. • `apic custom-theme:get-export-status` - Get the status of an export of a `custom theme`. • `apic custom-theme:get-import-status` - Get the status of an import of a `custom theme`. For more information, see Using the custom-theme commands.
`apic custom-translation`	Manage your `custom translations`.	• `apic custom-translation:create-export` - Create a task to export a `custom translation`. • `apic custom-translation:create-import` - Create a task to import a `custom translation`. • `apic custom-translation:delete-export` - Cancel any ongoing or recently created export tasks. • `apic custom-translation:delete-import` - Cancel any ongoing or recently created import tasks. • `apic custom-translation:get-export` - Streams an export artifact of a `custom translation` back to the provided task ID. • `apic custom-translation:get-export-status` - Get the status of an export of a `custom translation`. • `apic custom-translation:get-import-status` - Get the status of an import of a `custom translation`. For more information, see Using the custom-translation commands.
`apic custom-webserver-page`	Manage your `custom-webserver-page` content.	• `apic custom-webserver-page:delete` - Delete a custom HTML web server error page. • `apic custom-webserver-page:get` - Get the content of a custom HTML web server error page. • `apic custom-webserver-page:set` - Set a custom HTML web server error page by using a valid HTML5 content file. For more information, see Using the custom-webserver-page commands.
`apic drupal-config`	Manage your `drupal-config`.	• `apic drupal-config:delete` - Delete a Drupal config object, or a specific config key. • `apic drupal-config:get` - Get all of the key values for a Drupal config object, or a specific config key value. • `apic drupal-config:list` - List the Drupal config names by prefix, or list all. • `apic drupal-config:set` - Set a new Drupal config key and value, or update an existing config key and value. For more information, see Using the drupal-config commands.

Command	Description	Subcommands
`apic drupal-state`	Manage your `drupal-state` keys and values.	• `apic drupal-state:delete` - Delete a specific Drupal state key and value. • `apic drupal-state:get` - Get a specific Drupal state key value. • `apic drupal-state:set` - Set a new Drupal state key and value, or update an existing state key value. For more information, see Using the drupal-state commands.
`apic entity`	Manage your `entity`.	• `apic entity:count` - A count of entities from a site. For more information, see Using the entity commands.
`apic export-entity`	Manage the export of your content entities.	• `export-entity:create` - Create a new export entity, which is the container for the entity content that you want to export. • `export-entity:add-content` - Adds content to an existing export entity. • `export-entity:get` - Returns a list of the content of a specific export entity. • `export-entity:remove-content` - Removes certain content from a specific export entity. • `export-entity:delete` - Deletes a specific export entity. • `export-entity:launch` - Launches an export entity polling task that creates a .tgz file of all of the entities that are contained in a specific export entity. Can be run with a `--no-poll` option, in which case the task doesn't return a .tgz file, but just returns the task ID. • `export-entity:get-launch-export` - Streams the content of a specific completed `export-entity:launch` task to a .tgz file. • `export-entity:delete-launch-export` - Cancels a currently running `export-entity:launch` task, and deletes any related artifacts. • `export-entity:get-launch-export-status` - Returns the status of a specific `export-entity:launch` task. • `export-entity:list` - Returns a list of all of the export entities within a specific Developer Portal. Each export entity contains a defined list of all of the entity content that will be exported if `export-entity:launch` is run. For more information, see Using the export-entity commands.
`apic factory-reset`	Perform a factory reset of your Developer Portal.	• `apic factory-reset:delete` - Perform a factory reset of your Developer Portal, deleting the portal service and all of its associated Developer Portal sites. Warning: The **factory-reset** command is irreversible. You must ensure that you have backups created before running this command. For more information, see Using the factory-reset command.
`apic forums`	Manage your `forums`.	• `apic forums:disable` - disables the Forum module for a given site. • `apic forums:enable` - enables the Forum module for a given site. For more information, see Using the forums commands.
`apic ip-security-enabled`	Manage your `IP security`.	• `apic ip-security-enabled:update` - Enable or disable the IP security at the portal service level. For more information, see Using the ip-security-enabled command.
`apic modules`	Manage your `modules`.	• `apic modules:delete` - Delete `modules`. • `apic modules:disable` - Disable `modules`. • `apic modules:enable` - Enable `modules`. • `apic modules:list` - Get a list of your `modules`. For more information, see Using the modules commands.
`apic php-memory`	Manage your `php-memory`.	• `apic php-memory:list` - List the PHP memory limit on the platform. • `apic php-memory:update` - Set the PHP memory limit on the portal platform. For more information, see Using the php-memory commands.
`apic platforms`	Manage your `platforms`.	• `apic platforms:list` - List the platforms of a portal service. For more information, see Using the platforms commands.
`apic product`	Manage your `product`.	• `apic product:add-attachment` - Add an attachment for the given product. • `apic product:add-tag` - Add a tag for the given product. • `apic product:get` - Get a product from a portal service. • `apic product:get-document` - Get a specific entire product document from the portal of the given organization and catalog. • `apic product:list` - List the products on a portal service. • `apic product:set-icon` - Set an icon image for the given product. For more information, see Using the product commands.
`apic security`	Manage your `security`.	• `apic security:clear-bans` - Clear all of the banned user/IP addresses on a portal site. For more information, see Using the security command.
`apic service`	Manage your `service` operations.	• `apic service:list` - List the Portal service currently installed. For more information, see Using the service command.

Command	Description	Subcommands
`apic service-ip-allowlist`	Manage your `service-ip-allowlist`, which enables specific IP addresses to be exempt from being blocked by portal security checks.	• `apic service-ip-allowlist:add` - Add specific IP addresses to the `allowlist`. • `apic service-ip-allowlist:delete` - Delete all of the IP addresses from the `allowlist`. • `apic service-ip-allowlist:list` - List all of the IP addresses that are currently on the `allowlist`. • `apic service-ip-allowlist:remove` - Remove one or more specific IP addresses from the `allowlist`. For more information, see Using the service-ip-allowlist commands.
`apic site`	Manage your `site`.	• `site:cache-rebuild` - Rebuild the cache of a portal site. • `apic site:check` - Run a platform check against a portal site. • `apic site:state` - Obtain the current state of a portal site. • `apic site:login-link` - Get an admin login link for a specific portal site. • `site:create-export` - Create a task to export a .tgz archive file of your entire site configuration. • `site:create-import` - Create a task to import an archive of your entire site configuration. • `site:delete-export` - Cancel any currently running `site:create-export` tasks, and delete any related artifacts. • `site:delete-import` - Cancel any currently running `site:create-import` tasks, and delete any related artifacts. • `site:get-export-status` - Return the status of a specific export task. • `site:get-export` - Stream the content of a specific completed export task to a .tgz file. • `site:get-import-status` - Return the status of a specific import task. Important: If you want to import a site export configuration file, the export file must have been created on the same version of API Connect as the version that you want to import to. For more information, see Using the site commands.
`apic site-config`	Manage your `site-config`.	• `apic site-config:create-export` - Create an export of a `site config`. • `apic site-config:create-import` - Create an import of a `site config`. • `apic site-config:delete-export` - Delete an export of a `site config`. • `apic site-config:delete-import` - Delete an import of a `site config`. • `apic site-config:get-export` - Get an export of a `site config`. • `apic site-config:get-export-status` - Get the status of an export of a `site config`. • `apic site-config:get-import-status` - Get the status of an import of a `site config`. For more information, see Using the site-config commands.
`apic sites`	Manage your `sites`.	• `apic sites:check` - Run a platform check against all portal sites on a portal service. • `apic sites:list` - List the sites of a portal service. For more information, see Using the sites commands.
`apic themes`	Manage your `themes`.	• `apic themes:delete` - Delete `themes`. • `apic themes:disable` - Disable `themes`. • `apic themes:enable` - Enable `themes`. • `apic themes:list` - Get a list of your `themes`. • `apic themes:set-default` - Set the default theme of the portal. For more information, see Using the themes commands.
`apic twig`	Manage the `twig` debugging operations on a specific portal site.	• `apic twig:debug-enable` - Enable twig debugging for a portal site. • `apic twig:debug-disable` - Disable twig debugging for a portal site. • `apic twig:debug-status` - Get the current state of twig debugging for a specific site. For more information, see Using the twig commands.
`apic version`	Get the version of your API Connect toolkit.	• `apic version --mode portaladmin` - Get toolkit version.

- **Using the api commands**
  You can use the **api** commands to get and list the APIs on your Developer Portal service.
- **Using the apic-config commands**
  You can use the **apic-config** commands to get the apic config of your Developer Portal service.
- **Using the application commands**
  You can use the **application** commands to get and list the applications on your Developer Portal service.
- **Using the backups commands**
  You can use the **backups** commands to list the backups on your Developer Portal service.
- **Using the consumer-org commands**
  You can use the **consumer-org** commands to get and list the consumer orgs on your Developer Portal service.
- **Using the content commands**
  You can use the **content** commands to list, import, and export content from your Developer Portal site.
- **Using the custom-module commands**
  You can use the **custom-module** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.
- **Using the custom-theme commands**
  You can use the **custom-theme** commands to create export and import tasks, delete export and import tasks, get the status of an export or import task, and get an

export artifact, on your Developer Portal service.

- **Using the custom-translation commands**
  You can use the **custom-translation** commands to create an import or export task that will import or export a .tgz of custom translations for a Developer Portal site. You can also delete the export and import tasks, get the status of an export or import task, and get a custom translation export artifact.
- **Using the custom-webserver-page commands**
  You can use the **custom-webserver-page** commands to customize the web server error pages on your Developer Portal service.
- **Using the drupal-config commands**
  You can use the **drupal-config** commands to list, get the values of, set values, create new, and delete the Drupal configuration objects on your Developer Portal service.
- **Using the drupal-state commands**
  You can use the **drupal-state** commands to store and retrieve information about the state of your Developer Portal system.
- **Using the entity commands**
  You can use the **entity** commands to get the entity count on your Developer Portal service.
- **Using the export-entity commands**
  You can use the **export-entity** commands to manage the export of content entities from your Developer Portal site.
- **Using the factory-reset command**
  You can use the **factory-reset** command to perform a factory reset of your Developer Portal, which will delete the portal service and all of its associated Developer Portal sites.
- **Using the forums commands**
  You can use the **forums** commands to disable or enable the forums on your Developer Portal service.
- **Using the ip-security-enabled command**
  You can use the **ip-security-enabled** commands to toggle IP security on your Developer Portal. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. Banned IP addresses can be cleared by using the `security:clear-bans` command. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. Note that IP security is enabled by default on the Developer Portal.
- **Using the modules commands**
  You can use the **modules** commands to list, enable, disable and delete the modules on your Developer Portal service.
- **Using the php-memory commands**
  You can use the **php-memory** commands to list or update the PHP memory limits on your Developer Portal.
- **Using the platforms commands**
  You can use the **platforms** commands to list the platforms on your Developer Portal service.
- **Using the product commands**
  You can use the **product** commands to get and list the products on your Developer Portal service.
- **Using the security command**
  You can use the **security** command to clear all of the banned user/IP addresses on a Developer Portal site.
- **Using the service command**
  You can use the **service** command to list the Developer Portal service from the Developer Portal database.
- **Using the service-ip-allowlist commands**
  You can use the **service-ip-allowlist** commands to add, remove, and list the IP addresses on your Developer Portal `allowlist`. IP addresses that are on the `allowlist` are exempt from being blocked by Developer Portal security checks, for example, load balancer and proxy IPs.
- **Using the site commands**
  You can use the **site** commands to check the site on your Developer Portal service, and to export and import an entire site configuration.
- **Using the site-config commands**
  You can use the **site-config** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.
- **Using the sites commands**
  You can use the **sites** commands to check and list the sites on your Developer Portal service.
- **Using the themes commands**
  You can use the **themes** commands to list, enable, disable, delete, and set the default themes on your Developer Portal service.
- **Using the twig commands**
  You can use the **twig** commands to enable, disable, and check the state of twig debugging on a specific site on your Developer Portal service.

Using the api commands

You can use the **api** commands to get and list the APIs on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. List the APIs from a site.
   For example,

   ```
   apic --mode portaladmin api:list --org orgid/name --server management_server --catalog catalogid/name
   ```

   - `management_server` is the endpoint URL of the management server (required).
   - `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
   - `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
   - `format_type` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.
3. Get an API from a site by using the `id` or `name:version`.
   For example,

   ```
   apic --mode portaladmin api:get --org orgid/name --server management_server --catalog catalogid/name id/name:version
   ```

   - `id/name:version` - The ID or name:version of a specific API (required). For example, `id-of-api-called-example-3` or `example:3.0.0`.
4. Get a specific entire API document from the portal of the provided organization and catalog by using the `id` or `name:version`.

   ```
   apic --mode portaladmin api:get-document --org orgid/name --server management_server --catalog catalogid/name --format
   format_type  id/name:version
   ```

   For example,

   ```
   apic --mode portaladmin api:get-document --org ibm --server management_server --catalog portal-test --format yaml  intuiz-
   api:1.0.0
   ```

5. Add an attachment for your API within your Developer Portal.
   For example,

   ```
   apic --mode portaladmin api:add-attachment -s management_server --org orgid/name --catalog catalogid/name --
   attachment_name attachment.txt --attachment_description "API documentation" mortgage-management-api:1.0.0 ./attachment.txt
   Loading File (Large files may take a while)...

   Attachment successfully added to api mortgage-management-api:1.0.0.
     This product now has 4 attachments.
   ```

   - `attachment_name` is the name given to the attachment when it is uploaded (required).
   - `attachment_description` is the description of the attachment that is displayed to the users.
6. Set an icon for your API within your Developer Portal.
   For example,

   ```
   apic --mode portaladmin api:set-icon -s management_server --org orgid/name --catalog catalogid/name --icon_description
   "API icon" mortgage-management-api:1.0.0 ./icon.png
   Loading File (Large files may take a while)...

   Icon successfully set for api mortgage-management-api:1.0.0
   ```

   - `icon_description` is the icon description to be displayed to users and it is used as an alt text for the image (required).
7. Add a tag (Category) for your API within your Developer Portal.
   For example,

   ```
   apic --mode portaladmin api:add-tag -s management_server --org orgid/name --catalog catalogid/name --tag_name
   top_level_element/next_level_element/lower_level_element mortgage-management-api:1.0.0
   Successfully set taxonomy tag top_level_element/next_level_element/lower_level_element
     for api mortgage-management-api:1.0.0
   ```

   - `tag_name` is the tag name. For example, `top_level_element/next_level_element` (required).

Using the apic-config commands

You can use the **apic-config** commands to get the apic config of your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Get the apic config of a site.
   For example,

   ```
   apic --mode portaladmin apic-config:get --org orgid/name --server management_server --catalog name/id
   ```

Using the application commands

You can use the **application** commands to get and list the applications on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. List the applications on a site.
   For example,

   ```
   apic --mode portaladmin application:list --org orgid/name --server management_server --catalog name/id
   ```

3. Get an application from a site by using application id.
   For example,

   ```
   apic --mode portaladmin application:get --org orgid/name --server management_server --catalog name/id id
   ```

Using the backups commands

You can use the **backups** commands to list the backups on your Developer Portal service.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Get the availability zone of the portal service's backups that you want to list:

   ```
   apic availability-zones --org orgid/name --server management_server
   ```

3. List all the portal services in that availability zone

   ```
   apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
   ```

4. Get the selected portal service's name

   ```
   apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-
   NAME
   ```

5. List the backups of the portal service

```
apic --mode portaladmin backups:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

Using the consumer-org commands

You can use the **consumer-org** commands to get and list the consumer orgs on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. List the consumer orgs of a site.
   For example,

   ```
   apic --mode portaladmin consumer-org:list --org orgid/name --server management_server --catalog name/id
   ```

3. Get a consumer org from a site by using org id.
   For example,

   ```
   apic --mode portaladmin consumer-org:get --org orgid/name --server management_server --catalog name/id id
   ```

Using the content commands

You can use the **content** commands to list, import, and export content from your Developer Portal site.

About

The Developer Portal is based on the Drupal content management system. In Drupal, a content entity is an item of content data that is intended to be displayed to site users, such as text, images, and HTML markup. Content entities are grouped into entity types, and most of these entity types are also subdivided into entity subtypes, also known as bundles. The **content** commands enable you to list, import, and export either one type of content entity, or all of the content entity types, at a time. If you want to manage the export of assorted content entity types at the same time, you can use the **export-entity** commands; see Using the export-entity commands for more information.

Procedure

1. Log in as Provider org as seen in the following example.

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

   For full details of the `apic login` command, see Logging in to a management server.

For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. List the content entity types that are exportable from your site by using the following `content:list-types` command. This command returns a list of all of the content entity types that are allowed to be exported from your site. If a type isn't listed in the return, that type is either blocked or isn't a content entity type.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:list-types
```

Where:
- *`management_server`* is the endpoint URL of the management server (required).
- *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
- *`catalogid/name`* is the ID or name of the catalog that the site belongs to (required).

Each content entity type can have subtypes, which are known as bundles. These bundles can be used in the `content:list` command, when you want to list a particular bundle type of a content entity. In the following example, `page` is a bundle type of the content entity type `node`.

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog content:list-types
block_content:
  - basic
comment:
  - comment
  - comment_forum
export_entity: {  }
file: {  }
import_entity: {  }
media: {  }
menu_link_content: {  }
node:
  - article
  - blog_post
  - faq
  - forum
  - page
shortcut:
  - default
taxonomy_term:
  - forums
  - tags
```

3. List all of the entities in your site for a given content entity type and, if required, bundle by using the following `content:list` command.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:list --entity_type
entity_type_name --bundle bundle_name
```

Where:
- *`entity_type_name`* is the type of content entity that you want to list (required). You can use the `content:list-types` command to find the list of available content entity types.
- *`bundle_name`* is the name of the bundle type that you want to list (optional).

The following example lists the content of the `page` bundle type for the `node` content entity. Included in the return is an `Entity ID` for each item in the content list.

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog content:list --entity_type node --
bundle page
-
  Title: 'Terms of use'
  Langcode: en
  'Entity ID': '1'
  Bundle: page
  UUID: 75fc430e-fc4d-47e3-bc5f-7f23063716ac
-
  Title: 'Privacy Policy'
  Langcode: en
  'Entity ID': '2'
  Bundle: page
  UUID: ddff300b-16fe-4d1e-be6f-ce7040131b68
```

4. Export your site content entities by using the following `content:create-export` command. This command creates an export task that returns a .tgz file containing the content entities. By default, the command polls the status of the task every 30 seconds until the task is finished, at which point the export file is returned. You can export a specific content entity type, or specific entities within an entity type, or you can generate a .tar file that contains all of the exportable content from your site. Note that exporting all of your site content can take a long time to complete.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:create-export --
entity_type entity_type_name --entity_ids entity_ids
```

Where:
- *`entity_type_name`* is the type of content entity that you want to export (required). If you want to export all of the exportable content from your site, set this option to `all`.
- *`entity_ids`* is a comma separated list of the entity IDs that you want to export (optional).
- `--no-poll` is an optional flag meaning that the command won't poll the status of the task, but instead just returns the task ID. This task ID can then be used with the `content:get-export` command to return the export file.

Note: The command continues to poll until the maximum polling time of 25 minutes is reached. At which point you can use the commands `content:get_export_status` and `content:get-export` to get the status and, if the task has finished, an artifact.
The following example creates an export task for the `Terms of use` and `Privacy Policy` entity IDs of the `node` content entity.

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog content:create-export --
entity_type node --entity_ids 1,2
201 CREATED - Task ID: 8h86b8utjtw51

Response Code - 202:

Message(s) -
The status of this task is: RUNNING
```

```
Response Code - 200:

Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed

Saving File (Large files may take a while)...
Incoming project saved to content_createExport-20230627114023.tgz
```

The command saves the export file to your running directory. The export .tgz file includes a root.json file that contains an index of all of the entities that are contained in the .tgz file, as well as a *content_entity_type*.json file for each content entity type exported. The *content_entity_type*.json file contains all of the information needed to be able to import those entities into a different Developer Portal site.

5. If you ran the `content:create-export` command with the `--no-poll` option, you can use the following commands to manage your content export.
   - `content:delete-export`
     Cancels any currently running `content:create-export` tasks, and deletes any related artifacts. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:delete-export --
     task_id task_id
     ```

     Where ***task_id*** is the ID of the export task on the queue.
   - `content:get-export`
     Streams the content of a specific completed export task to a .tgz file. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:get-export --
     task_id task_id
     ```

   - `content:get-export-status`
     Returns the status of a specific export task. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:get-export-
     status --task_id task_id
     ```

6. Import your site content entities by using the `content:create-import` command. This command launches an import polling task that polls the status of the task every 30 seconds until the task is finished, at which point a summary of the import is returned. If the command is run with the `--no-poll` option, the task doesn't return the status of the task, but just returns the task ID.
   Important: Before you import any content, make sure that you backup the site that you're importing to, in case content gets overwritten that you weren't expecting.

   ```
   apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:create-import
   export_file_name
   ```

   Where:
   - ***export_file_name*** is the name of the export file that you want to import. You can import .tgz files that were created by using the `content:create-export` or `export-entity:launch` commands.
   - `--no-poll` is an optional flag meaning that the command won't poll the status of the task, but instead just returns the task ID. This task ID can then be used with the `content:get-import-status` command to get the status of the task.

   Note: The command continues to poll until the maximum polling time of 25 minutes is reached. At which point you can use the command `content:get-import-status` to get the status of the task.
   In the following example, the export file from Step 4 is imported into a different Developer Portal site on a different server:

   ```
   apic --mode portaladmin --server my_management_server2 --org my_prod_org --catalog my_prod_catalog content:create-import
   content_createExport-20230627114023.tgz
   Loading File (Large files may take a while)...

   201 CREATED - Task ID: zmaskh7oo6531tty

   Response Code - 202:

   Message(s) -
   The status of this task is: QUEUED

   Response Code - 202:

   Message(s) -
   The status of this task is: RUNNING

   Response Code - 202:

   Message(s) -
   The status of this task is: RUNNING

   Response Code - 200:

   Message(s) -
   2023-10-18 13:36:18: CLI task (content:import) starting.
   2023-10-18 13:36:29: [notice] [1/2] - [node.page:1] "Terms of use" - Updated
   (https://my_management_server2/my_prod_catalog/tsandcs)
    [notice] [2/2] - [node.page:2] "Privacy Policy" - Updated (https://my_management_server2/my_prod_catalog/privacy)
   2023-10-18 13:37:20: CLI task (content:import) completed successfully.
   ```

   Depending on the content imported, the following three types of import can be shown in the import summary messages:
   - `Overridden` - In this case a content entity existed with the same entity ID but not the same UUID, and there was also no reference to it in the synchronization table, so the entity was overridden. Care must be taken when importing entities across different sites until they are in sync with one another, as data can be lost. For example, say you have a blog post about APIs with an entity ID of `50` on a test site, and you wanted to import this post onto your production site. However, the production site already has a blog post with an entity ID of `50` that talks about policies, in which case that post would be overwritten by the blog post about APIs, and be given the imported UUID. The next time that entity ID of `50` is imported, the import type will be `Updated`, because the UUID is now the same.
   - `Updated` - In this case the same content entity, with the same UUID, was found, so its contents were updated with the content from the import file.
   - `Created` - In this case no content entity of this type was found, so a new entity was created, and it was populated with the data from the import file.

7. If you ran the `content:create-import` command with the `--no-poll` option, you can use the following commands to manage your content import.
   - `content:delete-import`
     Cancels any currently running `content:create-import` tasks, and deletes any related artifacts. For example:

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:delete-import --
task_id task_id
```

Where *task_id* is the ID of the import task on the queue.

- **content:get-import-status**
  Returns the status of a specific import task. For example:

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name content:get-import-
status --task_id task_id
```

Related reference

- Using the export-entity commands

Using the custom-module commands

You can use the **custom-module** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
For full details of the **apic login** command, see Logging in to a management server.

2. Export your custom modules.
The **custom-module:create-export** command creates an export task against the portal of the specified **catalog** and **org**. The command then polls the status of the task until it has **FINISHED** and the artifacts are streamed back.
Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-module:get-export-status** and **custom-module:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin custom-module:create-export --catalog name/id --org orgid/name --server management_server
```

- The module .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format **custom_module_createExport-*20200217134637*.tgz**.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin custom-module:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your custom modules.
The **custom-module:create-import** command uses the supplied .tgz file to import the custom modules into the portal of the specified **catalog** and **org**. The command then polls the status of the task until the task has either **FINISHED** successfully, or failed because of an error.
Note:
   - The higher level folders of the archive can contain only these characters, a-z and 0-9.
   - The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-module:get-import-status** to get the status of the import.
   - If you import a custom module that already exists you overwrite the existing custom module.

```
apic --mode portaladmin custom-module:create-import --catalog name/id --org orgid/name --server management_server
/PATH/TO/MODULE/TGZ
```

For example,

```
apic --mode portaladmin custom-module:create-import --catalog prod --org ibm --server api.productionexample.com
C:/users/example/desktop/custom_module_createExport-20200217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom modules are deleted.
For example,

```
apic --mode portaladmin custom-module:delete-export --catalog name/id --org orgid/name --server management_server --format
json --task_id string
```

```
apic --mode portaladmin custom-module:delete-import --catalog name/id --org orgid/name --server management_server --format
json --task_id string
```

Related information

- `apic custom-module`

Using the custom-theme commands

You can use the **custom-theme** commands to create export and import tasks, delete export and import tasks, get the status of an export or import task, and get an export artifact, on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Export your custom themes.
   The **custom-theme:create-export** command creates an export task against the portal of the specified `catalog` and `org`. The command then polls the status of the task until it has `FINISHED` and the artifacts are streamed back.
   Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-theme:get-export-status** and **custom-theme:get-export** to get the status and potentially an artifact.

   ```
   apic --mode portaladmin custom-theme:create-export --catalog name/id --org orgid/name --server management_server
   ```

   - The theme .tgz is saved to the directory in which you ran the command.
   - The saved .tgz has the format `custom_theme_createExport-20200217134637.tgz`.
   - You can save your exports into your change control management system.

   For example,

   ```
   apic --mode portaladmin custom-theme:create-export --catalog dev --org ibm --server api.stagingexample.com
   ```

3. Import your custom themes.
   The **custom-theme:create-import** command uses the supplied .tgz file to import the custom themes into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.
   Note:
   - The higher level folders of the archive can contain only these characters, a-z and 0-9.
   - The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-theme:get-import-status** to get the status of the import.
   - If you import a custom theme that already exists you overwrite the existing custom theme.

   ```
   apic --mode portaladmin custom-theme:create-import --catalog name/id --org orgid/name --server management_server
   /PATH/TO/theme/TGZ
   ```

   For example,

   ```
   apic --mode portaladmin custom-theme:create-import --catalog prod --org ibm --server api.productionexample.com
   C:/users/example/desktop/custom_theme_createExport-20200217134637.tgz
   ```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom themes are deleted.
   For example,

   ```
   apic --mode portaladmin custom-theme:delete-export --catalog name/id --org orgid/name --server management_server --format
   json --task_id string
   ```

   ```
   apic --mode portaladmin custom-theme:delete-import --catalog name/id --org orgid/name --server management_server --format
   json --task_id string
   ```

Related information

- `apic custom-theme`

Using the custom-translation commands

You can use the **custom-translation** commands to create an import or export task that will import or export a .tgz of custom translations for a Developer Portal site. You can also delete the export and import tasks, get the status of an export or import task, and get a custom translation export artifact.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Export your custom translations.
   The **custom-translation:create-export** command creates an export task against the portal of the specified `catalog` and `org`, to export an archive of the custom translations of the site. The command then polls the status of the task until it has `FINISHED` and the artifacts are streamed back. The task ID is printed to the terminal.
   Note:
   - The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-translation:get-export-status** and **custom-translation:get-export** to get the status and potentially an artifact.
   - The translation .tgz archive is saved to the directory in which you ran the command.
   - The saved .tgz has the format `custom_translation_createExport-20210217134637.tgz`.
   - You can save your exports into your change control management system.

   ```
   apic --mode portaladmin custom-translation:create-export --catalog catalogid/name --org orgid/name --server
   management_server
   ```

   - *`catalogid/name`* is the ID or name of the catalog that the site belongs to (required).
   - *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
   - *`management_server`* is the endpoint URL of the management server (required).
   - `--langcodes` is a comma separated list of the language codes, for example `es,zh-hans`. You can use this option to specify which custom translations to export.
   - `--no-poll` can be used to stop the command from polling the task, and just return the task ID.

   For example:

   ```
   apic --mode portaladmin custom-translation:create-export --catalog dev --org ibm --server api.stagingexample.com
   ```

3. Import your custom translations.
   The **custom-translation:create-import** command uses the supplied .tgz file to create a task to import the custom translations into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error. The task ID is printed to the terminal.
   Note:
   - The .tgz archive must contain only .po translation files, and nothing else.
   - The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-translation:get-import-status** to get the status of the import.
   - If you import a custom translation that already exists, the existing custom translation is overwritten.

   ```
   apic --mode portaladmin custom-translation:create-import --catalog catalogid/name --org orgid/name --server
   management_server /PATH/TO/MODULE/TGZ
   ```

   - `--no-poll` can be used to stop the command from polling the task, and just return the task ID.

   For example:

   ```
   apic --mode portaladmin custom-translation:create-import --catalog prod --org ibm --server api.productionexample.com
   C:/users/example/desktop/custom_translation_createExport-20210217134637.tgz
   ```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem. No custom translations are deleted.
   For example,

   ```
   apic --mode portaladmin custom-translation:delete-export --catalog catalogid/name --org orgid/name --server
   management_server --format json --task_id taskid
   ```

   ```
   apic --mode portaladmin custom-translation:delete-import --catalog catalogid/name --org orgid/name --server
   management_server --format json --task_id taskid
   ```

   - *`taskid`* is the ID of the import or export task that was created on the queue (required).
   - `--format` is the output format that can be seen if you run the command with debug. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

   For example:

   ```
   apic --mode portaladmin custom-translation:delete-import --catalog prod --org ibm --server api.productionexample.com  --
   task_id 4hythbptta6z9lyx
   ```

Using the custom-webserver-page commands

You can use the **custom-webserver-page** commands to customize the web server error pages on your Developer Portal service.

The **custom-webserver-page** commands enable you to set, get, and delete custom HTML content for the web server error pages of index, 404, 40x, and 50x. You can embed base64 encoded images, such as .svg and .png, or inline .svg images in your content, but the maximum size of the content of each error page is 8 MB.
Note:

- The custom web server error pages that can be set by using the **custom-webserver-page** commands, are the pages that are returned when the URL of a specific portal site isn't used. For example, if your portal site was https://example.com/myorg/dev, but you went to https://example.com/myorg/deb, then this is when a custom 404 error page is returned. However, if the requested URL matches a portal site, then the web server will use the Drupal error handlers, and will respond according to the Drupal configuration.
- All custom content that you set must conform to the HTML5 specification. If you try to set custom content that doesn't conform to HTML5, it will fail with the following error message:

  ```
  Error: The supplied custom HTML failed validation.
  ```

- Setting and deleting custom web server pages can take several minutes to complete, depending on the size of the content.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Set a custom HTML web server error page on the Developer Portal. The maximum size of the page is 8 MB, and you must ensure that the content conforms to the HTML5 specification, or the **custom-webserver-page:set** command will fail with a validation error.

   ```
   apic --mode portaladmin custom-webserver-page:set --server management_server --portal_service_name portal --page_type
   page_type filename.html
   ```

   - *`management_server`* is the endpoint URL of the management server (required).
   - *`portal`* is the name of the portal service (required).
   - *`page_type`* is the type of web server error page that you want to run the command against (required). Valid values are `index`, `404`, `40x`, or `50x`.
   - *`filename`* is the name of the HTML file that contains the content for the custom web server error page.

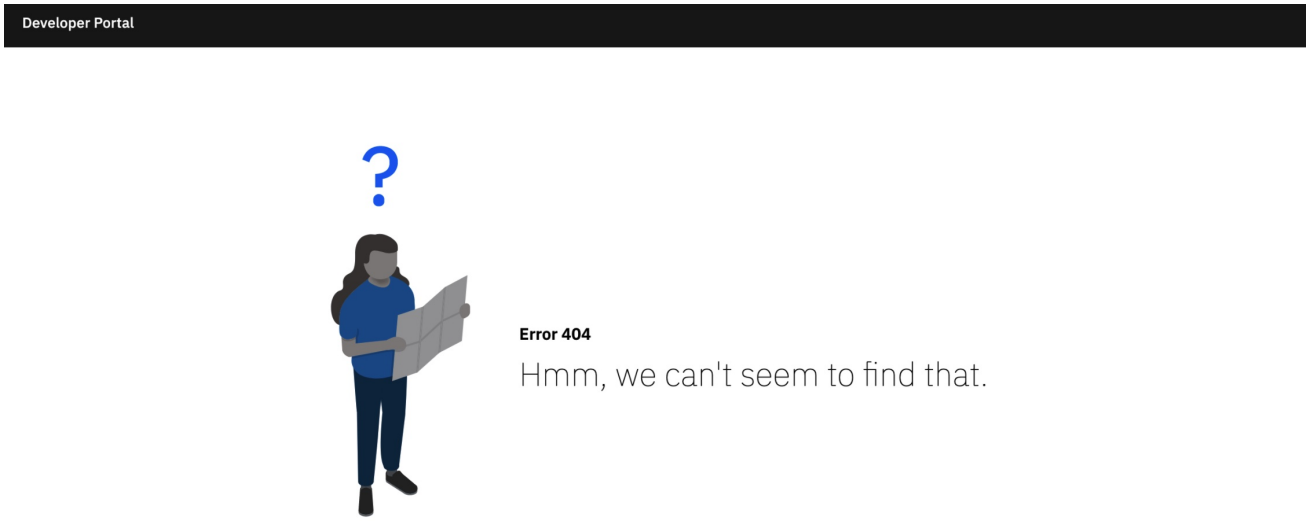
   For example, the following screen capture shows a standard 404 web server response:

   404 Not Found

   Then the following **custom-webserver-page:set** command is run:

   ```
   apic --mode portaladmin custom-webserver-page:set --server my.management.server.com --portal_service_name portal --
   page_type 404 custom-404.html
   Loading File (Large files may take a while)...

   Successfully added the custom HTML content for page type 404.
   ```

   The following custom 404 error page is now displayed:



3. Get the custom content of a specific page type and save it to a file named <page_type>-timestamp.html.

   ```
   apic --mode portaladmin custom-webserver-page:get --server management_server --portal_service_name portal --page_type
   page_type
   ```

   For example:

   ```
   apic --mode portaladmin custom-webserver-page:get --server my.management.server.com --portal_service_name portal --
   page_type 404
   Incoming project saved to 404-20211027121830.html
   ```

   If there is no custom content available for the page type, the following error is displayed:

   ```
   apic --mode portaladmin custom-webserver-page:get --server my.management.server.com --portal_service_name portal --
   page_type 50x
   Error: There is currently no custom HTML content for page type 50x.
   ```

4. Delete the custom content of a specific page type. If there is no custom content available for a page type, the error response defaults to the standard web server error page.

   ```
   apic --mode portaladmin custom-webserver-page:delete --server management_server --portal_service_name portal --page_type
   page_type
   ```

   For example:

   ```
   apic --mode portaladmin custom-webserver-page:delete --server my.management.server.com --portal_service_name portal --
   page_type 404
   The custom HTML content for page type 404 has been deleted.
   ```

Using the drupal-config commands

You can use the **drupal-config** commands to list, get the values of, set values, create new, and delete the Drupal configuration objects on your Developer Portal service.

For example, you can use the **drupal-config** commands to disable CSS and JS aggregation to help with twig debug, see Example.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. List all of the available Drupal config for a site.

```
apic --mode portaladmin drupal-config:list --server management_server --catalog catalogid/name --org orgid/name --format
format_type
```

- *management_server* is the endpoint URL of the management server (required).
- *catalogid/name* is the ID or name of the catalog that the site belongs to (required).
- *orgid/name* is the ID or name of the provider organization that the catalog belongs to (required).
- *format_type* is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

For example:

```
apic --mode portaladmin drupal-config:list --server my.management.server.com --org ibm --catalog portal-test --format yaml
admin_toolbar_tools.settings:
  name: admin_toolbar_tools.settings
adminimal_admin_toolbar.settings:
  name: adminimal_admin_toolbar.settings
advagg.settings:
  name: advagg.settings
advagg_css_minify.settings:
  name: advagg_css_minify.settings
advagg_js_minify.settings:
  name: advagg_js_minify.settings
autologout.settings:
  name: autologout.settings
...
```

Note that the example has been reduced as the full response is very large.

3. List a specific Drupal config for a site.

```
apic --mode portaladmin drupal-config:list --server management_server --catalog catalogid/name --org orgid/name --prefix
config_prefix --format format_type
```

- *config_prefix* is the prefix of the Drupal config, for example `system`. If omitted, the command returns all config names in the system.

For example:

```
apic --mode portaladmin drupal-config:list --server my.management.server.com --org ibm --catalog portal-test --format yaml
--prefix system.theme
system.theme:
  name: system.theme
system.theme.global:
  name: system.theme.global
```

4. Update a Drupal config key value.

```
apic --mode portaladmin drupal-config:set -server management_server --catalog catalogid/name --org orgid/name  --
config_name config_name --config_key config_key --config_value config_value
```

- *config_name* is the name of the Drupal config object, for example `system.site` (required).
- *config_key* is the Drupal config key, for example `page.front` (required).
- *config_value* is the value to assign to the config key (required).

For example, to set the default theme to `claro`:

```
apic --mode portaladmin drupal-config:set --server my.management.server.com --org ibm --catalog portal-test --config_name
system.theme --config_key default --config_value claro
// Do you want to update default key in system.theme config?: yes.
```

5. Create a new Drupal config key and set the value.

```
apic --mode portaladmin drupal-config:set -server management_server --catalog catalogid/name --org orgid/name  --
config_name config_name --config_key new_config_key --config_value new_config_value
```

For example, to create a new config key and set the value in the `system.site` config object:

```
apic --mode portaladmin drupal-config:set --server my.management.server.com --org ibm --catalog portal-test --config_name
system.site --config_key new_key --config_value new_value
// new_key key does not exist in system.site config. Do you want to create a new config key?: yes.
```

6. Get all of the key values for a Drupal config object.

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name  --format
format_type --config_name config_name
```

For example, to find out what themes are available:

```
apic --mode portaladmin drupal-config:get --server my.management.server.com --org ibm --catalog portal-test --format json
--config_name system.theme
{
    "admin": "seven",
    "default": "claro",
    "_core": {
        "default_config_hash": "fOjer9hADYYnbCJVZMFZIIM1azTFWyg84ZkFDHfAbUg"
    },
    "defaultplus": "bartisk"
}
```

7. Get the value of a specific key for a Drupal config object.

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name  --format
format_type --config_name config_name --config_key config_key
```

For example, to find out what the default theme is:

```
apic --mode portaladmin drupal-config:get --server my.management.server.com --org ibm --catalog portal-test --format json
--config_name system.theme --config_key default
{
```

```
    "system.theme:default": "claro"
}
```

8. Delete a Drupal config object, or a specific config key and its value.

```
apic --mode portaladmin drupal-config:delete --server management_server --catalog catalogid/name --org orgid/name  --
config_name config_name --config_key config_key
```

   - *config_key* is the Drupal config key, for example `page.front` (optional).

   For example, to delete the config key `new_key` and its value:

```
apic --mode portaladmin drupal-config:delete --server my.management.server.com --org ibm --catalog portal-test --
config_name system.site --config_key new_key
The config object/value was successfully deleted.
```

Example

The following scenario gives an example of how to turn off CSS and JS aggregation to help with twig debug. However, on a production system aggregation should be enabled for performance reasons.

1. Check the settings for performance, which include settings for CSS and JS aggregation:

```
apic --mode portaladmin drupal-config:get --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance
```

2. Turn off CSS aggregation:

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance --config_key css.preprocess --config_value 0
```

3. Turn off JS aggregation:

```
apic --mode portaladmin drupal-config:set --server management_server --catalog catalogid/name --org orgid/name --
config_name system.performance --config_key js.preprocess --config_value 0
```

4. Then, rebuild the cache:

```
apic --mode portaladmin site:cache-rebuild  --server management_server --catalog catalogid/name --org orgid/name
```

When loading the site in a browser, you should now see CSS and JS aggregation is disabled when using development tools for site debug or customization.
Note:

- Disabling aggregation affects site performance, so if the site goes into production remember to turn aggregation back on again.
- Although the settings are shown as `true` or `false`, to set them by using the `drupal-config:set` command, you must apply a value of `1` for `true`, and `0` for `false`.

Using the drupal-state commands

You can use the **drupal-state** commands to store and retrieve information about the state of your Developer Portal system.

Information about the state of your Developer Portal system is stored in the Developer Portal database, and therefore this information is lost if the database is dropped, or the Developer Portal site is re-installed from configuration.

The **drupal-state** commands can be used to set, update, get, or delete a state key value. For example, you can use the **drupal-state** commands to enable and disable maintenance mode for your Developer Portal; for more information, see the scenario How to enable and disable maintenance mode on your Developer Portal.

Warning: Use the **drupal-state** command with caution. This command enables you to modify keys for your Developer Portal site state. Incorrect usage of this command can bring down your Developer Portal, and then require restoring it from a backup.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

   For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Create a new Drupal state key, or update an existing key, and set the value.

```
apic --mode portaladmin drupal-state:set -server management_server --catalog catalogid/name --org orgid/name  --
input_format input_format_type --state_key state_key_name --state_value state_value
```

Where:

- `management_server` is the endpoint URL of the management server (required).
- `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
- `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
- `input_format_type` is the input format of the value for the state key. Can be `string`, `integer`, `float`, `boolean`, `json`, or `yaml`. Defaults to `auto`, which means that you can omit this parameter, and the toolkit CLI will decide which format to use.
- `state_key_name` is the name of the state key, for example `system.cron_last` (required).
- `state_value` is the value to assign to the state key (required).
- `--format` is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

For example, to create a new key called `testKey`, and set the value to `123`:

```
apic --mode portaladmin drupal-state:set -s my.management.server.com -o myorg -c dev-catalog --input_format yaml --
state_key testKey --state_value 123
Successfully set state testKey to the value of 123
```

3. Get the value for a particular Drupal state key.

```
apic --mode portaladmin drupal-state:get --server management_server --catalog catalogid/name --org orgid/name  --format
format_type --state_key state_key_name
```

For example, to get the value of the state key `testKey`:

```
apic --mode portaladmin drupal-state:get -s my.management.server.com -c dev-catalog -o myorg --format yaml --state_key
testKey
testKey: 123
```

4. Delete a specific Drupal state key and its value.

```
apic --mode portaladmin drupal-state:delete --server management_server --catalog catalogid/name --org orgid/name --
state_key state_key_name
```

For example, to delete the state key `testKey` and its value:

```
apic --mode portaladmin drupal-state:delete -s my.management.server.com -c dev-catalog -o myorg --state_key testKey
The key/value was successfully deleted.
```

Related tasks

- Toggling the site in and out of maintenance mode

Using the entity commands

You can use the **entity** commands to get the entity count on your Developer Portal service.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

2. Get the entity count for a site.
For example,

```
apic --mode portaladmin entity:count --org orgid/name --server management_server --catalog name/id
```

Using the export-entity commands

You can use the **export-entity** commands to manage the export of content entities from your Developer Portal site.

About

The Developer Portal is based on the Drupal content management system. In Drupal, a content entity is an item of content data that is intended to be displayed to site users, such as text, images, and HTML markup. Content entities are grouped into entity types, and most of these entity types are also subdivided into entity subtypes, also known as bundles. The **export-entity** commands enable you to create a predefined list, called an export entity, of all of the content entities that you want to export. You can create multiple export entities, each one containing its own set of assorted entity types. Export entities persist, so you can add content and remove content as required.

If you want to export only one type of content entity, or all of the content entity types, at a time, you can use the **content** commands; see Using the content commands for more information. You also use these commands when you want to import your entity content.

Procedure

1. Log in as Provider org as seen in the following example.

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.

   For full details of the `apic login` command, see Logging in to a management server.

   For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. Create an export entity by using the following `export-entity:create` command. This command creates a container for the entity content that you want to export.

   ```
   apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:create --
   export_entity_name "export_entity_name"
   ```

   Where:
   - *management_server* is the endpoint URL of the management server (required).
   - *orgid/name* is the ID or name of the provider organization that the catalog belongs to (required).
   - *catalogid/name* is the ID or name of the catalog that the site belongs to (required).
   - *export_entity_name* is the name of the export entity that you want to create. Note that this name is not a unique identifier.

   The command returns an ID for the newly created export entity. For example:

   ```
   apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:create --
   export_entity_name "Main Content"
   [notice] Created new export entity Main Content, id 5
   ```

   Make a separate note of the ID number that is returned, as you will need this ID for future commands.
3. List all of the export entities that are available for your site by using the following `export-entity:list` command.

   ```
   apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:list
   ```

   The command returns a list of the available export entities, including their ID, and the date and time that they were created. For example:

   ```
   apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:list
   -
     id: '5'
     name: 'Main Content'
     created_time: '2023-09-12 15:05:04'
   ```

4. Add content to an export entity by using the following `export-entity:add-content` command.

   ```
   apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:add-content --
   export_entity_id export_entity_id --entity_type entity_type_name --entity_ids entity_ids
   ```

   Where:
   - *export_entity_id* is the ID of the export entity (required).
   - *entity_type_name* is the type of content entity that you want to add to the export entity (required). You can use the `content:list-types` command to find out the type of content entities that are exportable from your site; see Using the content commands.
   - *entity_ids* is a comma separated list of the entity IDs that you want to add to the export entity (optional). If you want to add all of the content entities for a particular entity type, you don't need to set any entity IDs.

   The following example shows three different types of entity content being added to the export entity `5`:

   ```
   apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:add-content --
   export_entity_id 5 --entity_type node --entity_ids 1,2
   [notice] Successfully added entity type node - "1, 2" to export entity 5

   apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:add-content --
   export_entity_id 5 --entity_type block_content --entity_ids 1,2
   [notice] Successfully added entity type block_content - "1, 2" to export entity 5
   ```

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:add-content --
export_entity_id 5 --entity_type shortcut --entity_ids 1
[notice] Successfully added entity type shortcut - "1" to export entity 5
```

5. Delete content from an export entity by using the following **export-entity:remove-content** command.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:remove-content
--export_entity_id export_entity_id --entity_type entity_type_name --entity_ids entity_ids
```

Where:
- ***export_entity_id*** is the ID of the export entity (required).
- ***entity_type_name*** is the type of content entity that you want to remove (required).
- ***entity_ids*** is a comma separated list of the entity IDs that you want to remove (optional). If you want to remove all of the content entities for a particular entity type, you don't need to set any entity IDs.

For example, to remove entity ID **1** of the content entity type **shortcut** from the export entity **5**, you can run the following command:

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:remove-content --
export_entity_id 5 --entity_type shortcut --entity_ids 1
[notice] Successfully removed entity type shortcut - "1" from export entity 5
```

6. View the content details of an export entity by using the following **export-entity:get** command. This command returns a list of all of the content entities in a specific export entity ID, and is the entity content that will be exported if the **export-entity:launch** command is run on that ID.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:get --
export_entity_id export_entity_id
```

In the following example, export entity **5** contains four different content entities:

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:get --
export_entity_id 5
- bundle: basic
  id: "1"
  title: No Apps Found [en]
  type: block_content
- bundle: page
  id: "1"
  title: Terms of use
  type: node
- bundle: basic
  id: "2"
  title: No Apis Found [en]
  type: block_content
- bundle: page
  id: "2"
  title: Privacy Policy
  type: node
```

7. Export your content entities by running the following **export-entity:launch** command. This command launches an export entity polling task that creates a .tgz file of all of the entities that are contained in a specific export entity. By default, the command polls the status of the task every 30 seconds until the task is finished, at which point the export file is returned. If the command is run with the **--no-poll** option, the task doesn't return a .tgz file, but just returns the task ID.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:launch --
export_entity_id export_entity_id
```

Where:
- **--no-poll** is an optional flag meaning that the command won't poll the status of the task, but instead just returns the task ID. This task ID can then be used with the **export-entity:get-export** command to return the export file.

Note: The command continues to poll until the maximum polling time of 25 minutes is reached. At which point you can use the commands **export-entity:get_export_status** and **export-entity:get-export** to get the status and, if the task has finished, an artifact.
The **export-entity:launch** command can be run multiple times, and the content of the export entity updated as required by using the **export-entity:add-content** and **export-entity:remove-content** commands.

The following example exports the contents of export entity **5**:

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog export-entity:launch --
export_entity_id 5
201 CREATED - TASK ID: saleo5261k5a2y

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 200:

Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed

Saving File (Large files may take a while)...
Incoming project saved to export_entity_launchExport-20230912165
```

The command saves the export to your running directory. The export .tgz file includes a root.json file that contains an index of all of the entities that are contained in the .tgz file, as well as a *content_entity_type*.json file for each content entity type exported. The *content_entity_type*.json file contains all of the information needed to be able to import those entities into a different Developer Portal site.

8. If you ran the **export-entity:launch** command with the **--no-poll** option, you can use the following commands to manage your entity export.
   - **export-entity:delete-launch-export**
     Cancels any currently running **export-entity:launch** tasks, and deletes any related artifacts. For example:

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:delete-launch-export --task_id task_id
```

Where *task_id* is the ID of the export task on the queue.

- **export-entity:get-launch-export**
  Streams the content of a specific completed **export-entity:launch** task to a .tgz file. For example:

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:get-launch-export --task_id task_id
```

- **export-entity:get-launch-export-status**
  Returns the status of a specific **export-entity:launch** task. For example:

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name export-entity:get-launch-export-status --task_id task_id
```

What to do next

You can import your export entity archive files by using the **content** commands; see Using the content commands for more information.

Related reference

- Using the content commands

Using the factory-reset command

You can use the **factory-reset** command to perform a factory reset of your Developer Portal, which will delete the portal service and all of its associated Developer Portal sites.

Warning: The **factory-reset** command is irreversible. You must ensure that you have backups created before running this command.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
   ```

   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
   For full details of the **apic login** command, see Logging in to a management server.

2. Output a dry run status of a Developer Portal factory reset:

   ```
   apic --mode portaladmin factory-reset:delete --server management_server --portal_service_endpoint portal_endpoint
   ```

   - *management_server* is the endpoint URL of the management server.
   - *portal_endpoint* is the URL endpoint of the portal service.

   For example:

   ```
   apic --mode portaladmin factory-reset:delete --server api.stagingexample.com --portal_service_endpoint
   https://api.portal.mystack.com
   ```

3. Perform a factory reset of the Developer Portal, deleting the portal service and all of its associated Developer Portal sites:
   Warning: The **factory-reset** command is irreversible. You must ensure that you have backups created before running this command.

   ```
   apic --mode portaladmin factory-reset:delete --server management_server --portal_service_endpoint portal_endpoint --execute_reset true
   ```

   - **--execute_reset** must be set to **true** to trigger the Developer Portal reset.

   For example:

   ```
   apic --mode portaladmin factory-reset:delete --server api.stagingexample.com --portal_service_endpoint
   https://api.portal.mystack.com  --execute_reset true
   ```

Using the forums commands

You can use the **forums** commands to disable or enable the forums on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Disable the Forum module on your Developer Portal site.
   This command deletes all of the taxonomy terms in the forum vocabulary, and then disables the Forum module.

   ```
   apic --mode portaladmin forums:disable --org orgid/name --server management_server --catalog name/id
   ```

   - *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
   - *`management_server`* is the endpoint URL of the management server (required).
   - *`name/id`* is the ID or name of the catalog that the site belongs to (required).

   For example,

   ```
   apic --mode portaladmin forums:disable --org ibm --server api.stagingexample.com --catalog dev
   ```

3. Enable the Forum module on your Developer Portal site.

   ```
   apic --mode portaladmin forums:enable --org orgid/name --server management_server --catalog name/id
   ```

   For example,

   ```
   apic --mode portaladmin forums:enable --org ibm --server api.stagingexample.com --catalog dev
   ```

Using the ip-security-enabled command

You can use the **ip-security-enabled** commands to toggle IP security on your Developer Portal. When IP security is enabled, modules such as the Drupal Perimeter Defence module, or flood control, will block client IP addresses suspected of malicious behavior, as expected. Banned IP addresses can be cleared by using the `security:clear-bans` command. You might want to turn off IP security if you are performing penetration tests, or if you cannot pass through the client IP address from your external load balancer. Note that IP security is enabled by default on the Developer Portal.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Enable IP security on the Developer Portal:

   ```
   apic --mode portaladmin ip-security-enabled:update --server management_server --portal_service_name portal --enabled true
   ```

   - *`management_server`* is the endpoint URL of the management server.
   - *`portal`* is the name of the portal service.
   - `--enabled` is set to `true` to enable IP security. Can be set to `false` to disable IP security.

   For example:

   ```
   apic --mode portaladmin ip-security-enabled:update --server my.management.server.com --portal_service_name
   my_portal_service --enabled true
   IP security has been successfully set to true
   ```

3. Disable IP security on the Developer Portal:

```
apic --mode portaladmin ip-security-enabled:update --server management_server --portal_service_name portal --enabled false
```

Using the modules commands

You can use the **modules** commands to list, enable, disable and delete the modules on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

   For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. List the modules from a *packagename*.
   For example,

   ```
   apic --mode portaladmin modules:list --org orgid/name --server management_server --catalog name/id --package
   packagename1,packagename2
   ```

3. Enable the *modulename* module. You can upload this module by using the **custom-module:create-import** command. For more information, see Using the custom-module commands.
   For example,

   ```
   apic --mode portaladmin modules:enable --org orgid/name --server management_server --catalog name/id --modules modulename
   ```

   - `management_server` is the endpoint URL of the management server (required).
   - `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
   - `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
   - `--package` enables you to filter by project packages. You can use multiple comma separated values.
   - `--status` enables you to filter by extension status. Can be `_ enabled`, `disabled` and/or `'not installed'`. You can use multiple comma separated values.
   - `--custom` enables you to filter your results to only the modules that you have installed on your Developer Portal site.

4. Disable the *modulename* module. If you disable a module, you must also disable all its dependencies.
   For example,

   ```
   apic --mode portaladmin modules:disable --org orgid/name --server management_server --catalog name/id --modules
   modulename,module_lib
   ```

5. Delete the *modulename* module. The module and all its dependencies must be disabled before you can delete them.
   For example,

   ```
   apic --mode portaladmin modules:delete --org orgid/name --server management_server --catalog name/id --modules modulename
   ```

Note that if you want to manage the Forum module, you can use the `forums` commands. For more information, see Using the forums commands.

Using the php-memory commands

You can use the **php-memory** commands to list or update the PHP memory limits on your Developer Portal.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   ```

```
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For full details of the **apic login** command, see Logging in to a management server.

2. Get the availability zone of the portal service's sites that you want to list:

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone:

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name:

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
```

5. List the current PHP memory limits on the Developer Portal platform:

```
apic --mode portaladmin php-memory:list --server management_server --portal_service_name PORTAL-SERVICE-NAME
```

You receive output like this example:

```
Current web memory is set to: 768M
Current admin memory is set to: 768M
```

6. Update the current PHP memory limits on the Developer Portal platform:

```
apic --mode portaladmin php-memory:update --server management_server --portal_service_name PORTAL-SERVICE-NAME --memory_value 1024
```

You receive output like this example:

```
Setting php memory limit to 1024M within the database
Setting php memory limit to 1024M in /var/devportal/storenosync/php.ini
Setting php memory limit to 1024M in /web/local/php_config/php.ini
Done
```

Note: Consider the memory resource limits that are allocated to your Developer Portal pods before you set this value. Setting this value too high can result in the portal pods restarting because the containers request more memory than is allowed by Kubernetes. Setting the value too low can result in the internal portal processes failing because of lack of available memory.

Using the platforms commands

You can use the **platforms** commands to list the platforms on your Developer Portal service.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
For full details of the **apic login** command, see Logging in to a management server.

2. Get the availability zone of the portal service's platforms that you want to list:

```
apic availability-zones --org orgid/name --server management_server
```

3. List all the portal services in that availability zone

```
apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
```

4. Get the selected portal service's name

```
apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
```

5. List the platforms of the portal service

```
apic --mode portaladmin platforms:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

Using the product commands

You can use the **product** commands to get and list the products on your Developer Portal service.

1. Log in as Provider org as seen in the following example.

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. Get a list of products from a site.

For example,

```
apic --mode portaladmin product:list --org orgid/name --server management_server --catalog catalogid/name
```

- *management_server* is the endpoint URL of the management server (required).
- *catalogid/name* is the ID or name of the catalog that the site belongs to (required).
- *orgid/name* is the ID or name of the provider organization that the catalog belongs to (required).
- *format_type* is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.

3. Get a product from a site by using the *id* or *name:version*.

For example,

```
apic --mode portaladmin product:get --org orgid/name --server management_server --catalog catalogid/name id/name:version
```

- *id/name:version* - The ID or name:version of a specific product (required). For example, `id-of-product-called-example-3` or `example:3.0.0`.

4. Get a specific entire product document from the portal of the provided organization and catalog by using the *id* or *name:version*.

```
apic --mode portaladmin product:get-document --org orgid/name --server management_server --catalog catalogid/name --format
format_type  id/name:version
```

For example,

```
apic --mode portaladmin product:get-document --org ibm --server my.management.server.com --catalog portal-test --format
yaml  intuiz-product:1.0.0
```

5. Add an attachment for your product within your Developer Portal.

For example,

```
apic --mode portaladmin product:add-attachment -s management_server --org orgid/name --catalog catalogid/name --
attachment_name attachment.txt --attachment_description "Product documentation" awesome-sleek-soft-chips-product:1.0.0
./attachment.txt
Loading File (Large files may take a while)...

Attachment successfully added to product awesome-sleek-soft-chips-product:1.0.0.
  This product now has 2 attachments.
```

- *attachment_name* is the name given to the attachment when it is uploaded (required).
- *attachment_description* is the description of the attachment that is displayed to the users.

6. Set an icon for your product within your Developer Portal.
For example,

```
apic --mode portaladmin product:set-icon -s management_server --org orgid/name --catalog catalogid/name --icon_description
"product icon" awesome-sleek-soft-chips-product:1.0.0 ./icon.png
Loading File (Large files may take a while)...

Icon successfully set for product awesome-sleek-soft-chips-product:1.0.0
```

- *icon_description* is the icon description to be displayed to users and it is used as an alt text for the image (required).

7. Add a tag (Category) for your product within your Developer Portal.
For example,

```
apic --mode portaladmin product:add-tag -s management_server--org orgid/name --catalog catalogid/name --tag_name
top_level_element/next_level_element/lower_level_element awesome-sleek-soft-chips-product:1.0.0
Successfully set taxonomy tag top_level_element/next_level_element/lower_level_element
  for product api-connect-cassin-llc-awesome-sleek-soft-chips-product:1.0.0
```

- ***tag_name*** is the tag name. For example, ***top_level_element/next_level_element*** (required).

Using the security command

You can use the **security** command to clear all of the banned user/IP addresses on a Developer Portal site.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is **default-idp-2**.
   For full details of the **apic login** command, see Logging in to a management server.

2. Clear all of the user and IP addresses that are currently banned on a particular portal site:

   ```
   apic --mode portaladmin security:clear-bans --server management_server --catalog catalogid/name --org orgid/name
   ```

   - ***management_server*** is the endpoint URL of the management server.
   - ***catalogid/name*** is the ID or name of the catalog that the site belongs to.
   - ***orgid/name*** is the ID or name of the provider organization that the catalog belongs to.

   For example:

   ```
   apic --mode portaladmin security:clear-bans --server my.management.server.com --catalog portal-test --org ibm
   [notice] All bans cleared.
   ```

Note: You can also manage banned IP addresses by using the Developer Portal UI. For more information, see Managing banned IP addresses.

Using the service command

You can use the **service** command to list the Developer Portal service from the Developer Portal database.

1. Log in as Cloud manager:

   ```
   apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
   ```

   You can determine which identity provider to use in the **--realm** parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
   ```

   For example:

   ```
   apic identity-providers:list --scope admin --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: Cloud Manager User Registry
       realm: admin/default-idp-1
     - title: Corporate LDAP user registry
       realm: admin/corporate-ldap
   ```

   The **title** value should enable you to determine which identity provider to use; you can then copy the corresponding **--realm** parameter directly from the displayed **realm** value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is **default-idp-1**.
   For full details of the **apic login** command, see Logging in to a management server.

   For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. Get the availability zone of the portal service's sites that you want to list.

   ```
   apic availability-zones --org orgid/name --server management_server
   ```

   - **apic service:list** - List the Portal service currently installed.

- `-s, --server` *mgmt_endpoint_url*. Use the value of the platform API endpoint URL for this entry.
- `--org` *orgid*
- `--catalog` *catalog id*.
- `--debug`. If you need debug information.

3. List the installed Developer Portal service.

```
apic --mode portaladmin service:list -s management_server --portal_service_name PORTAL-SERVICE-NAME --format json
```

Using the service-ip-allowlist commands

You can use the **service-ip-allowlist** commands to add, remove, and list the IP addresses on your Developer Portal `allowlist`. IP addresses that are on the `allowlist` are exempt from being blocked by Developer Portal security checks, for example, load balancer and proxy IPs.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username admin --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.
For full details of the `apic login` command, see Logging in to a management server.

2. Add one or more IP addresses to the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:add --server management_server --portal_service_name portal --ips
"list_of_ips"
```

- *`management_server`* is the endpoint URL of the management server.
- *`portal`* is the name of the portal service.
- *`list_of_ips`* is a comma separated list of the IP addresses that you want to add to the `allowlist`.

For example:

```
apic --mode portaladmin service-ip-allowlist:add --server my.management.server.com --portal_service_name my_portal_service
--ips "123.456.34.56,123.456.46.67"
Successfully added IP: 123.456.34.56.
Successfully added IP: 123.456.46.67.
```

3. Delete all of the IP addresses that are currently on the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:delete --server management_server --portal_service_name portal
```

4. List all of the IP addresses that are currently on the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:list --server management_server --portal_service_name portal
```

5. Remove one or more IP addresses from the `allowlist`:

```
apic --mode portaladmin service-ip-allowlist:remove --server management_server --portal_service_name portal --ips
"list_of_ips"
```

Using the site commands

You can use the **site** commands to check the site on your Developer Portal service, and to export and import an entire site configuration.

About

You can use the **site** commands to check the health of a portal site, rebuild the cache of a site, and display the login link for an admin account of a site. You can also use them to export the configuration settings for an entire Developer Portal site, and import those settings into another site. For example, you could use the export and import commands to replicate the configuration of a test site into a production site.
Important: If you want to import a site export configuration file, the export file must have been created on the same version of API Connect as the version that you want to import to.

1. Log in as Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. Check the site on your portal service. This command checks the file system, database, and API Manager on your portal site, which might identify platform-related problems on your portal site.

```
apic --mode portaladmin site:check  --server management_server --catalog catalogid/name --org orgid/name --format
format_type
```

   - *`management_server`* is the endpoint URL of the management server (required).
   - *`catalogid/name`* is the ID or name of the catalog that the site belongs to (required).
   - *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
   - *`format_type`* is the output format. Can be `json`, `yaml`, `go-template=...`, `go-template-file=...`. Defaults to `yaml`.
3. Rebuild the cache of a specific portal site.

```
apic --mode portaladmin site:cache-rebuild  --server management_server --catalog catalogid/name --org orgid/name
```

   For example:

```
apic --mode portaladmin site:cache-rebuild --server management_server --org ibm --catalog portal-test
[success] Cache rebuild complete.
```

4. Display the state of a portal site associated with a particular organization and catalog. This command is useful as a general health check of the site.

```
apic --mode portaladmin site:state --server management_server --catalog catalogid/name --org orgid/name --format
format_type
```

```
apic --mode portaladmin site:state --server my.management.server.com --org ibm --catalog portal-test --format json
{
    "alias": "ibm.portal-test.my.management.server.com",
    "check_state": "OK: 200",
    "content_refresh_required": false,
    "known_to_apim": 1,
    "platform": "platform_devportal_9_x_10_99_99_99_20210922_2000",
    "snapshot_first_requested": "0000-00-00 00:00:00",
    "snapshots_failed_processing_since_last_success": 0,
    "snapshots_requested_since_last_received": 0,
    "state": "INSTALLED",
    "subscribed_to_webhooks": true,
    "url": "my.management.server.com/ibm/portal-test",
    "uuid": "cc0ad766-e372-4988-9d45-744eb8d381de.1cd9dc41-b08d-xyz"
}
```

5. Log in to your Developer Portal admin with a link. This command displays a one time login link for the admin account for your Developer Portal site that is specified by the catalog and organization.

```
 apic --mode portaladmin site:login-link  --server management_server --org ibm --catalog portal-test
sample output:
https://my.portal.server.com/ibm/portal-test/user/reset/1/12345678901/ABCde1f2gH3ijKL-
4M_NOPqrs5TU6VWxy7ZABCDEFg7h8/new/login
```

6. Export your entire site configuration by using the following `site:create-export` command. This command creates an export task that returns a .tgz file containing the site configuration. By default, the command polls the status of the task every 10 seconds until the task is finished, at which point the export file is returned. Note that exporting all of your site configuration can take a long time to complete.

```
apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:create-export
```

   Where:
   - `--no-poll` is an optional flag meaning that the command won't poll the status of the task, but instead just returns the task ID. This task ID can then be used with the `site:get-export` command to return the export file.

   Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands `site:get_export_status` and `site:get-export` to get the status and, if the task has finished, an artifact.
   For example:

```
apic --mode portaladmin --server my_management_server --org my_org --catalog my_catalog site:create-export
201 CREATED - Task ID: vfg6b8utjtw51

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 202:
```

```
Message(s) -
The status of this task is: RUNNING

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 200:

Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed

Saving File (Large files may take a while)...
Incoming project saved to site_createExport-20231013154023.tgz
```

The command saves the export file to your running directory. The export .tgz file includes a site-content.tgz file, and a site-config.tgz file. The export file will also include a site-custom-themes.tgz, or a site-custom-modules.tgz file, or both, depending on whether the site contains any custom themes or custom modules.

7. If you ran the **site:create-export** command with the **--no-poll** option, you can use the following commands to manage your site export.
   - **site:delete-export**
     Cancels any currently running **site:create-export** tasks, and deletes any related artifacts. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:delete-export --
     task_id task_id
     ```

     Where ***task_id*** is the ID of the export task on the queue.
   - **site:get-export**
     Streams the content of a specific completed export task to a .tgz file. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:get-export --
     task_id task_id
     ```

   - **site:get-export-status**
     Returns the status of a specific export task. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:get-export-status -
     -task_id task_id
     ```

8. Import your site configuration by using the **site:create-import** command. This command launches an import polling task that polls the status of the task every 10 seconds until the task is finished, at which point a summary of the import is returned. If the command is run with the **--no-poll** option, the task doesn't return the status of the task, but just returns the task ID.
   Important: If you want to import a site export configuration file, the export file must have been created on the same version of API Connect as the version that you want to import to.

   ```
   apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:create-import
   export_file_name
   ```

   Where:
   - ***export_file_name*** is the name of the export file that you want to import. You can import .tgz files that were created by using the **site:create-export** command.
   - **--no-poll** is an optional flag meaning that the command won't poll the status of the task, but instead just returns the task ID. This task ID can then be used with the **site:get-import-status** command to get the status of the task.

   Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **site:get-import-status** to get the status of the task.
   In the following example, the export file from Step 6 is imported into a production Developer Portal site on a different server:

```
apic --mode portaladmin --server my_management_server2 --org my_prod_org --catalog my_prod_catalog site:create-import
site_createExport-20231013154023.tgz
Loading File (Large files may take a while)...

201 CREATED - Task ID: 1z8e9zthfkhwdxh2

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 200:

Message(s) -
2023-10-11 10:50:05: CLI task (site:import) starting.
2023-10-11 10:50:08: Performing a whole site import
2023-10-11 10:50:08: 1. Importing Site's Configuration
2023-10-11 10:50:41: 2. Importing Site's Content
2023-10-11 10:51:30: [notice] [1/30] - [node.page:1] "Terms of use" - Updated (https://portal.ibm.com/apim-portal-
org2/test-google/tsandcs)
```

```
 [notice] [2/30] - [node.page:2] "Privacy Policy" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/privacy)
 [notice] [3/30] - [node.faq:3] "Using our APIs" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/node/3)
 [notice] [4/30] - [node.faq:4] "Apps" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/node/4)
 [notice] [5/30] - [node.page:5] "AsyncAPI" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/asyncapi)
 [notice] [6/30] - [node.page:6] "Kafka" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/asyncapi-kafka)
 [notice] [7/30] - [node.page:7] "MQ" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/asyncapi-mq)
 [notice] [8/30] - [node.article:17] "This is an article" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/node/17)
 [notice] [9/30] - [node.article:20] "article 2" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/node/20)
 [notice] [10/30] - [node.page:23] "Basic page 1" - Updated (https://portal.ibm.com/apim-portal-org2/test-google/node/23)
 [notice] [11/30] - [block_content.basic:1] "No Apps Found [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/1)
 [notice] [12/30] - [block_content.basic:2] "No Apis Found [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/2)
 [notice] [13/30] - [block_content.basic:3] "No Products Found [en]" - Updated (https://portal.ibm.com/apim-portal-
org2/test-google/admin/content/block/3)
 [notice] [14/30] - [block_content.basic:4] "No Blogs Found [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/4)
 [notice] [15/30] - [block_content.basic:5] "Welcome Banner [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/5)
 [notice] [16/30] - [block_content.basic:6] "Getting Started [en]" - Updated (https://portal.ibm.com/apim-portal-
org2/test-google/admin/content/block/6)
 [notice] [17/30] - [block_content.basic:7] "Get Help [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/7)
 [notice] [18/30] - [block_content.basic:8] "Forum Sidebar [en]" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/content/block/8)
 [notice] [19/30] - [menu_link_content:1] "Home" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/1/edit)
 [notice] [20/30] - [menu_link_content:2] "API Products" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/2/edit)
 [notice] [21/30] - [menu_link_content:3] "Apps" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/3/edit)
 [notice] [22/30] - [menu_link_content:4] "Forums" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/4/edit)
 [notice] [23/30] - [menu_link_content:5] "Sign in" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/5/edit)
 [notice] [24/30] - [menu_link_content:6] "Create account" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/6/edit)
 [notice] [25/30] - [menu_link_content:7] "Contact" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/7/edit)
 [notice] [26/30] - [menu_link_content:8] "Terms and Conditions" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/8/edit)
 [notice] [27/30] - [menu_link_content:9] "Privacy Policy" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/structure/menu/item/9/edit)
 [notice] [28/30] - [shortcut.default:1] "Add content" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/config/user-interface/shortcut/link/1)
 [notice] [29/30] - [shortcut.default:2] "All content" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/admin/config/user-interface/shortcut/link/2)
 [notice] [30/30] - [taxonomy_term.forums:1] "General discussion" - Updated (https://portal.ibm.com/apim-portal-org2/test-
google/taxonomy/term/1)
2023-10-11 10:52:38: CLI task (site:import) completed successfully.
```

The message section provides detailed information about the import as the configuration is being imported.

9. If you ran the `site:create-import` command with the `--no-poll` option, you can use the following commands to manage your content import.
   - `site:delete-import`
     Cancels any currently running `site:create-import` tasks, and deletes any related artifacts. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:delete-import --
     task_id task_id
     ```

     Where *`task_id`* is the ID of the import task on the queue.
   - `site:get-import-status`
     Returns the status of a specific import task. For example:

     ```
     apic --mode portaladmin --server management_server --org orgid/name --catalog catalogid/name site:get-import-status -
     -task_id task_id
     ```

Related reference

- Using the custom-module commands
- Using the custom-theme commands
- Using the site-config commands

Related information

- **`apic site`**

Using the site-config commands

You can use the **site-config** commands to create exports and imports, delete exports and imports, get the status of an export or import, and get an export, on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username cloud_username --password
   cloud_password
   ```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

2. Export your site configs.
The **site-config:create-export** command creates an export task against the portal of the specified `catalog` and `org`. The command then polls the status of the task until it has `FINISHED` and the artifacts are streamed back.
Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **site-config:get-export-status** and **site-config:get-export** to get the status and potentially an artifact.

```
apic --mode portaladmin site-config:create-export --catalog name/id --org orgid/name --server management_server
```

- The site config .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format `site_config_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin site-config:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Import your site configs.
The **site-config:create-import** command uses the supplied .tgz file to import the site configs into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.
Note: The higher level folders of the archive can contain only these characters, a-z and 0-9.
Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **site-config:get-import-status** to get the status of the import.

```
apic --mode portaladmin site-config:create-import --catalog name/id --org orgid/name --server management_server
/PATH/TO/SITE_CONFIG/TGZ
```

For example,

```
apic --mode portaladmin site-config:create-import --catalog prod --org ibm --server api.productionexample.com
C:/users/example/desktop/site_config_createExport-20200217134637.tgz
```

4. You can run the delete command to cancel any ongoing or recently created import or export tasks. Any temporarily created artifacts will be deleted from the portal filesystem.
For example,

```
apic --mode portaladmin site-config:delete-export --catalog name/id --org orgid/name --server management_server --format
json --task_id string
```

```
apic --mode portaladmin site-config:delete-import --catalog name/id --org orgid/name --server management_server --format
json --task_id string
```

Using the sites commands

You can use the **sites** commands to check and list the sites on your Developer Portal service.

1. Log in as Cloud manager:

```
apic login --server management_server --realm admin/identity_provider --username cloud_username --password cloud_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope admin --server mgmt_endpoint_url --fields title,realm
```

For example:

```
apic identity-providers:list --scope admin --server myserver.com --fields title,realm
total_results: 2
results:
  - title: Cloud Manager User Registry
    realm: admin/default-idp-1
  - title: Corporate LDAP user registry
    realm: admin/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default Cloud Manager Local User Registry for login as a member of the cloud administration organization is `default-idp-1`.
For full details of the `apic login` command, see Logging in to a management server.

2. Get the availability zone of the portal service's sites that you want to list.

   ```
   apic availability-zones --org orgid/name --server management_server
   ```

3. List all the portal services in that availability zone.

   ```
   apic portal-services:list --org orgid/name --server management_server --availability-zone availability_zone
   ```

4. Get the selected portal service's name.

   ```
   apic portal-services:get --org orgid/name --server management_server --availability-zone availability_zone PORTAL-SERVICE-NAME
   ```

5. Check the sites on your portal service. This command checks the file system, database, and API Manager on all your portal sites, which might identify platform-related problems on one or more of your portal sites.

   ```
   apic --mode portaladmin sites:check --server management_server --portal_service_name PORTAL-SERVICE-NAME --format yaml
   ```

6. List the sites of the portal service.

   ```
   apic --mode portaladmin sites:list --server management_server --portal_service_name PORTAL-SERVICE-NAME --format json
   ```

Using the themes commands

You can use the **themes** commands to list, enable, disable, delete, and set the default themes on your Developer Portal service.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

   For a summary of the general-purpose commands and their use, see Developer Portal CLI commands.

2. List the themes from a *packagename*.
   For example,

   ```
   apic --mode portaladmin themes:list --org orgid/name --server management_server --catalog catalogid/name --package packagename1,packagename2
   ```

   - `management_server` is the endpoint URL of the management server (required).
   - `catalogid/name` is the ID or name of the catalog that the site belongs to (required).
   - `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
   - `--package` enables you to filter by project packages. You can use multiple comma separated values.
   - `--status` enables you to filter by extension status. Can be `enabled`, `disabled` and/or `'not installed'`. You can use multiple comma separated values.
   - `--custom` enables you to filter your results to only the themes that you have installed on your Developer Portal site.

3. Enable the *themename* theme. You can upload this theme by using the **custom-theme:create-import** command. For more information, see Using the custom-theme commands.
   For example,

   ```
   apic --mode portaladmin themes:enable --org orgid/name --server management_server --catalog catalogid/name --themes themename
   ```

   - `--themes` is the name of the theme, and can be multiple comma separated values (required).

4. Disable the *themename* theme.
   For example,

   ```
   apic --mode portaladmin themes:disable --org orgid/name --server management_server --catalog catalogid/name --themes themename,theme_lib
   ```

5. Delete the *themename* theme. The theme and all its dependencies must be disabled before you can delete them.
   For example,

   ```
   apic --mode portaladmin themes:delete --org orgid/name --server management_server --catalog catalogid/name --themes themename
   ```

6. Set the *themename* theme to be the default theme. The theme must exist and be enabled for it to be set as a default theme.
   For example,

```
apic --mode portaladmin themes:set-default --org orgid/name --server management_server --catalog catalogid/name themename
```

- *themename* is the name of the theme to set as default.

Using the twig commands

You can use the **twig** commands to enable, disable, and check the state of twig debugging on a specific site on your Developer Portal service.

Note: Twig debugging should not be left enabled on production sites, as it causes performance issues, and can affect the caching on the site.

1. Log in as Provider org:

   ```
   apic login --server management_server --realm provider/identity_provider --username provider_username --password
   provider_password
   ```

   You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

   ```
   apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
   ```

   For example,

   ```
   apic identity-providers:list --scope provider --server myserver.com --fields title,realm
   total_results: 2
   results:
     - title: API Manager User Registry
       realm: provider/default-idp-2
     - title: Corporate LDAP user registry
       realm: provider/corporate-ldap
   ```

   The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
   For full details of the `apic login` command, see Logging in to a management server.

2. Enable twig debugging for a portal site.

   ```
   apic --mode portaladmin twig:debug-enable --server management_server --org orgid/name --catalog catalogid/name
   ```

   - `management_server` is the endpoint URL of the management server (required).
   - `orgid/name` is the ID or name of the provider organization that the catalog belongs to (required).
   - `catalogid/name` is the ID or name of the catalog that the site belongs to (required).

   For example:

   ```
   apic --mode portaladmin twig:debug-enable --server my.management.server.com --org ibm --catalog portal-test
   Setting twig debug to true for site my.management.server.com/ibm/portal-test.
   Rebuilding cache for my.management.server.com/ibm/portal-test so twig debug changes can take affect.
   ```

3. Disable twig debugging for a portal site.

   ```
   apic --mode portaladmin twig:debug-disable --server management_server --org orgid/name --catalog catalogid/name
   ```

   For example:

   ```
   apic --mode portaladmin twig:debug-disable --server my.management.server.com --org ibm --catalog portal-test
   Setting twig debug to false for site my.management.server.com/ibm/portal-test.
   Rebuilding cache for my.management.server.com/ibm/portal-test so twig debug changes can take affect.
   ```

4. Get the current state of twig debugging for a specific site.

   ```
   apic --mode portaladmin twig:debug-status --server management_server --org orgid/name --catalog catalogid/name
   ```

   For example:

   ```
   apic --mode portaladmin twig:debug-status --server my.management.server.com --org ibm --catalog portal-test
   The current state of twig debugging is true for my.management.server.com/ibm/portal-test.
   ```

Scenarios that use the Portal CLI

Here are some scenarios of how you can use the Portal commands in the toolkit CLI to accomplish tasks that are more complex to complete, or not possible in the Developer Portal UI.

- **Exporting and importing custom themes and site configuration**
  You can use the portal commands in the toolkit CLI to replicate changes from one Developer Portal instance to another. You might want to use these commands to copy your themes or a configuration change, from a staging environment to a production environment.
- **How to enable and disable maintenance mode on your Developer Portal**
  You can use the Portal commands in the toolkit CLI to enable and disable maintenance mode on your Developer Portal site.

Exporting and importing custom themes and site configuration

You can use the portal commands in the toolkit CLI to replicate changes from one Developer Portal instance to another. You might want to use these commands to copy your themes or a configuration change, from a staging environment to a production environment.

Before you begin

To complete this scenario, you must have a staging Developer Portal and a production Developer Portal enabled. You must have administrator access, and the permission `settings:manage`, to both to complete this scenario. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this scenario

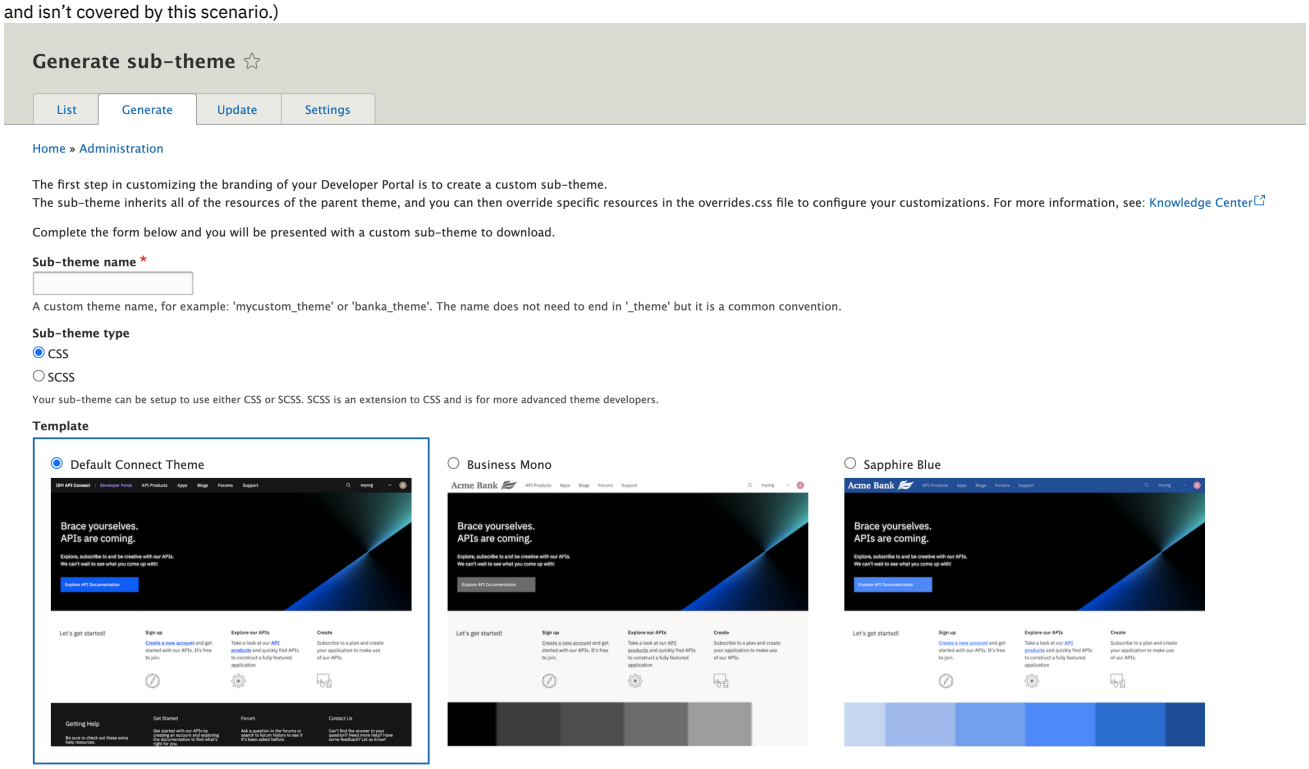
In this scenario, on your staging Developer Portal, you create a theme and make some configuration changes. Then, you use the toolkit CLI to replicate the changes on your production Developer Portal.

This scenario takes you through the following steps:

1. Create a custom theme on the staging Developer Portal.
2. Add a field to an application content type to change the `site-config` on the staging Developer Portal.
3. Use the `cli` to export the custom themes and `site-config` on the staging Developer Portal.
4. Use the `cli` to import the custom themes and `site-config` on the production Developer Portal.

Create a custom theme on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Appearance > Generate sub-theme. The `Generate sub-theme` window is displayed.
4. Enter a Sub-theme name, and select CSS for the Sub-theme type. (If you prefer, you can select SCSS, but this extension to CSS is for advanced theme developers, and isn't covered by this scenario.)



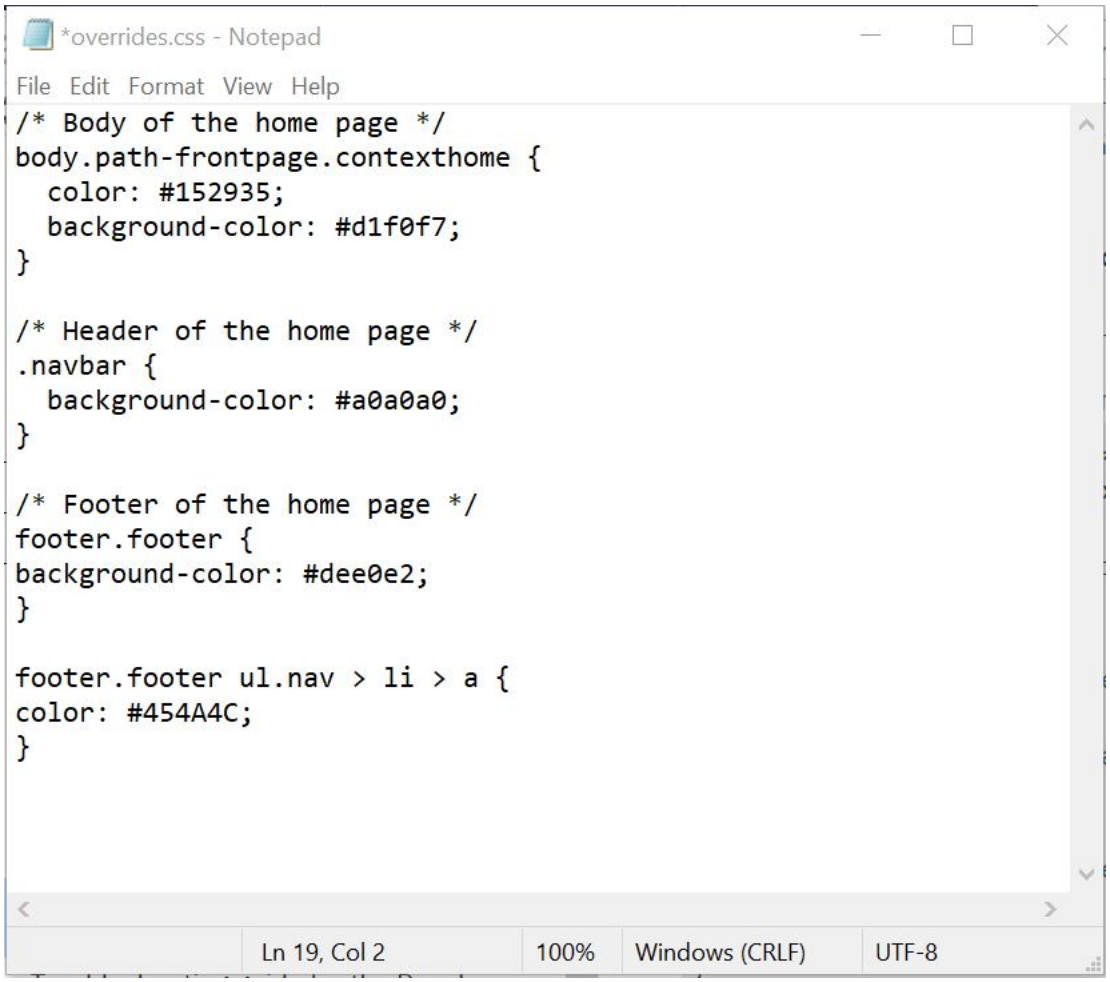
5. Select the Default template to base your `sub-theme` on. You can create a `sub-theme` based on a color template. However, for this scenario you use the default connect_theme template.
6. Click Generate.
7. Download the generated `sub-theme` to a location of your choice, and extract all the files from the .zip file.

On your computer, navigate to the files that you extracted. Find the overrides.css file in here banka_theme/banka_theme/css/overrides.css. You can now customize the overrides.css file.

1. Open the overrides.css file in your chosen editor.
2. Customize your `sub-theme` by entering the following elements into the overrides.css file:

```
/* Body of the home page */
body.path-frontpage.contexthome {
  color: #152935;
  background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
  background-color: #a0a0a0;
}
```

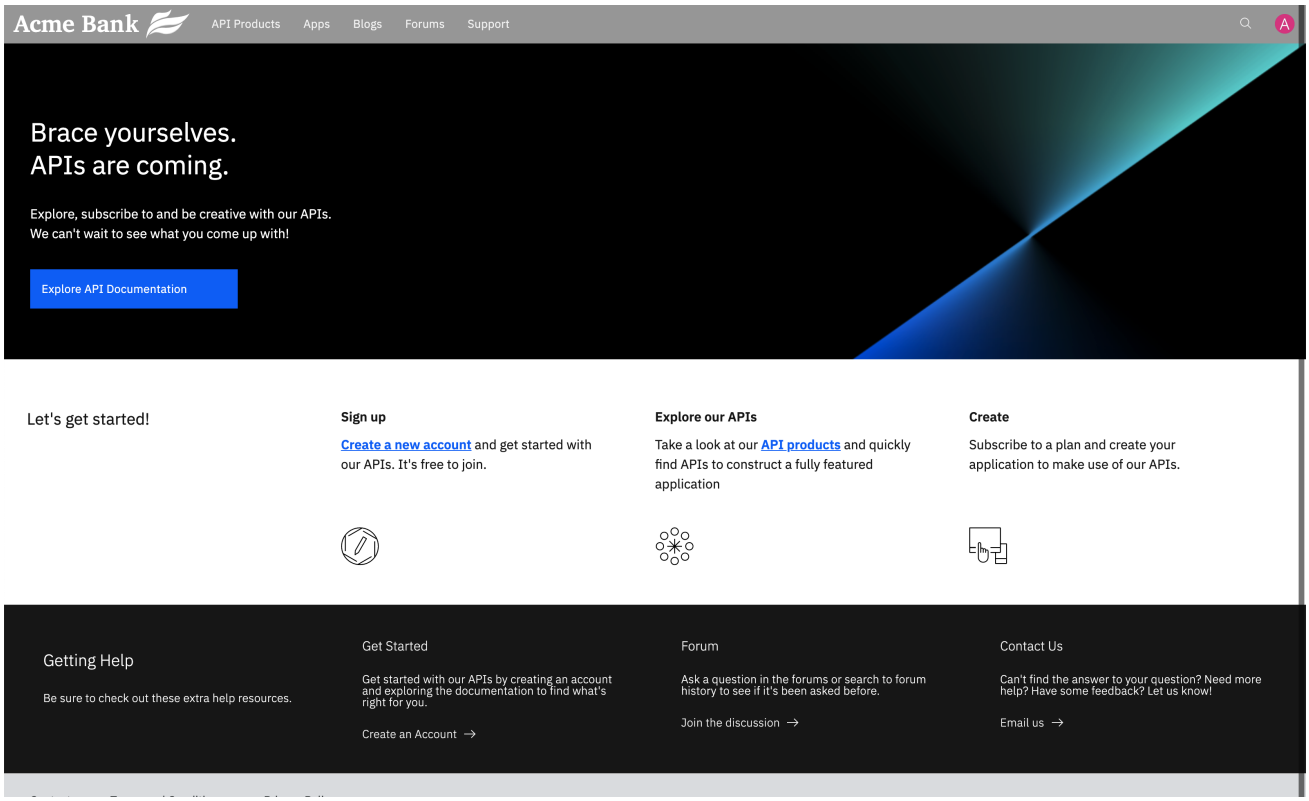
```
/* Footer of the home page */
footer.footer {
background-color: #dee0e2;
}

footer.footer ul.nav > li > a {
color: #454A4C;
}
```



3. Save the overrides.css file.

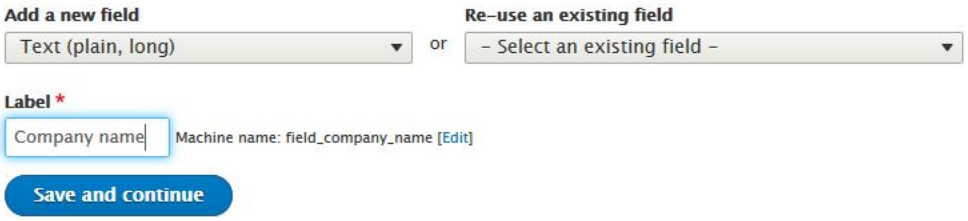
You can now install and enable your customized theme.

1. After you finish updating the overrides.css file, compress all the theme files back into the .zip **sub-theme** file that you downloaded originally.
2. In the Developer Portal, click Appearance > Install new theme. The **Install new theme** window is displayed.
3. In Upload a module or theme archive to install click Browse, and navigate to your newly updated compressed theme file.
4. Click Install to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of **Disabled themes**. Click Enable and set as default to set your new custom **sub-theme** as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



Add a field to an application content type to change the `site-config` on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Application > Manage fields.
4. Click + Add field.
5. In **`Add a new field`**, select Text (plain, long). Enter `Company name` as the label title.



6. Click Save and continue.
7. Leave the **`Allowed number of values`** set to 1 and click Save field settings.
8. Click Save field settings. You can now see your new field.

Use the toolkit CLI to export the custom themes and `site-config` on the staging Developer Portal

1. Log in to the management server.

   `apic login --server` ***`management_server`*** **`--realm`** ***`realm`*** **`--username`** ***`org_username`*** **`--password`** ***`org_password`***

   For example,

   `apic login -s api.stagingexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123£`

   For more information about variable options, see Logging in to a management server.

2. Export your custom themes.
   The **custom-theme:create-export** command creates an export task against the portal of the specified **`catalog`** and **`org`**. The command then polls the status of the task until it has **`FINISHED`** and the artifacts are streamed back.
   Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the commands **custom-theme:get-export-status** and **custom-theme:get-export** to get the status and potentially an artifact.

   `apic --mode portaladmin custom-theme:create-export --catalog` ***`name/id`*** **`--org`** ***`name/id`*** **`--server`** ***`management_server`***

- The theme .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format `custom_theme_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin custom-theme:create-export --catalog dev --org ibm --server api.stagingexample.com
```

3. Export your site config.
   The **site-config:create-export** command creates an export task against the portal of the specified `catalog` and `org`, the command then polls the status of the task until it has `FINISHED` and the artifacts are streamed back.

```
apic --mode portaladmin site-config:create-export --catalog name/id --org name/id --server management_server
```

- The site config .tgz is saved to the directory in which you ran the command.
- The saved .tgz has the format `site_config_createExport-20200217134637.tgz`.
- You can save your exports into your change control management system.

For example,

```
apic --mode portaladmin site-config:create-export --catalog dev --org ibm --server api.stagingexample.com
```

Use the toolkit CLI to import the custom themes and `site-config` on the production Developer Portal

1. Log in to the management server.

```
apic login --server management_server --realm realm --username org_username --password org_password
```

For example,

```
apic login -s api.productionexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123£
```

For more information about variable options, see Logging in to a management server.

2. Import your custom themes.
   The **custom-theme:create-import** command uses the supplied .tgz file to import the custom themes into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.
   Note: The command continues to poll until the maximum polling time of 15 minutes is reached. At which point you can use the command **custom-theme:get-import-status** to get the status of the import.

```
apic --mode portaladmin custom-theme:create-import --catalog name/id --org name/id --server management_server
/PATH/TO/THEME/TGZ
```

For example,

```
apic --mode portaladmin custom-theme:create-import --catalog prod --org ibm --server api.productionexample.com
C:/users/example/desktop/custom_theme_createExport-20200217134637.tgz
```

3. Import your site config.
   The **site-config:create-import** command uses the supplied .tgz file to import the site config into the portal of the specified `catalog` and `org`. The command then polls the status of the task until the task has either `FINISHED` successfully, or failed because of an error.

```
apic --mode portaladmin site-config:create-import --catalog name/id --org name/id --server management_server
/PATH/TO/CONFIG/TGZ
```

For example,

```
apic --mode portaladmin site-config:create-import --catalog prod --org ibm --server api.productionexample.com
C:/users/example/desktop/site_config_createExport-20200217134637.tgz
```

What you did in this scenario

In this scenario, you created a theme and changed the configuration on your staging Developer Portal. Then, you used the toolkit CLI to replicate the changes on to your production Developer Portal.

You can now see the changes that you made on your production Developer Portal.

- **Exporting and importing custom themes and site configuration by using the Portal Admin API**
  The platform REST APIs for IBM® API Connect provide complete access to the capability of the platform. So, you can use the toolkit CLI to replicate changes from one Developer Portal instance to another. However, if the toolkit CLI was unavailable for some reason, you can use the Portal Admin API.

Exporting and importing custom themes and site configuration by using the Portal Admin API

The platform REST APIs for IBM® API Connect provide complete access to the capability of the platform. So, you can use the toolkit CLI to replicate changes from one Developer Portal instance to another. However, if the toolkit CLI was unavailable for some reason, you can use the Portal Admin API.

Before you begin

To complete this scenario, you must have a staging Developer Portal and a production Developer Portal enabled. You must have administrator access, or the permission `settings:manage`, to both to complete this scenario. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

Also, you must have downloaded and installed the toolkit. For more information, see Installing the toolkit.

About this scenario

In this scenario, you use the toolkit to obtain some authentication. Then, on your staging Developer Portal, you create a theme and make some configuration changes. Then, you use the Portal Admin API to replicate the changes on your production Developer Portal.

This scenario takes you through the following steps:

1. Obtaining the client ID and secret, and a bearer token.
2. Create a custom theme on the staging Developer Portal.
3. Add a field to an application content type to change the `site-config` on the staging Developer Portal.
4. Use the Portal Admin API to export the custom themes and `site-config` on the staging Developer Portal.
5. Use the Portal Admin API to import the custom themes and `site-config` on the production Developer Portal.

Obtaining the client ID and secret, and a bearer token.

From your toolkit:

1. Obtaining the client ID and secret.
   a. Obtain the list of administrator scoped providers to determine the correct realm to use:

      ```
      apic identity-providers:list --scope admin --server <management_server>
      ```

      For example,

      ```
      apic identity-providers:list --scope admin --server api.stagingexample.com
      ```

      Here is an example of some possible output:

      ```
      default-idp-1
      ```

   b. Run `apic login`, complete the details:

      ```
      apic login --server management_server --realm cloud_realm --username admin_username --password admin_password
      ```

      For example,

      ```
      apic login -s api.stagingexample.com --realm provider/default-idp-1 --username admin --password Qwerty123£
      ```

   c. Log in to cloud manager, and register an app. Make a new file locally and call it `app1.json`. Then, copy into it the following code snippet:

      ```
      {
        "name": "app1",
        "client_id": "app1id",
        "client_secret": "app1secret",
        "client_type": "toolkit"
      }
      ```

   d. Run the registration create command:

      ```
      apic registrations:create --server management_server app1.json
      ```

      For example,

      ```
      apic registrations:create --server api.stagingexample.com app1.json
      ```

      Here is an example of some possible output:

      ```
      app1    [state: enabled]   https://api.stagingexample.com/api/cloud/registrations/d2b7572f-3406-42a7-bdcc-
      85f177860df7
      ```

2. Now, you can get a bearer token from the provider org.
   a. To obtain a list of provider realms:

      ```
      apic identity-providers:list --scope provider --server management_server
      ```

      For example,

      ```
      apic identity-providers:list --scope provider --server api.stagingexample.com
      ```

      Here is an example of some possible output:

      ```
      default-idp-2
      ```

   b. Run `apic login`, complete the details:

      ```
      apic login --server management_server --realm provider_realm --username org_username --password org_password
      ```

      For example,

      ```
      apic login -s api.stagingexample.com --realm provider/default-idp-2 --username ibm --password Qwerty123£
      ```

   c. Create an HTTP request to obtain an access token:

      ```
      curl -k -X POST -d '{"username": "org_username", "password": "org_password", "realm": "provider/default-idp-2",
      "client_id": "app_id", "client_secret": "app_secret", "grant_type": "password"}' -H 'Content-Type: application/json'
      -H 'Accept: application/json' https://management_server/api/token
      ```

      For example,

```
curl -k -X POST -d '{"username": "steve", "password": "T****8", "realm": "provider/default-idp-2", "client_id":
"app1id", "client_secret": "app1secret", "grant_type": "password"}' -H 'Content-Type: application/json' -H 'Accept:
application/json' https://api.stagingexample.com/api/token
```

Here is an example of some possible output:


```
{
    "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjBDQkI2Mjg5MkU4QUNFNEIzNDZCQzcyNDMxMUY0QzE4NTU2MUE4NjkifQ.eyJqdGkiOiJlN
DY2M2U4MC0wMzlkLTRiNDUtYmFjZC04NjA2MDcwZDIwODIiLCJuYW1lc3BhY2UiOiJiMTU4YjdhYy1kMTEzLTQxNDctODE1OS05ZDliNGViZjE3ZGE6Zj
dhY2IwNzMtYzU4ZC00ZjhkLWIwZDYtMGNlM2YwYjIyYmNmOmZjMWRlOTE2LWI1NDAtNDE4MS1iZTJiLWM4ZGU0NGVlYTMzZiIsImF1ZCI6Ii9hcGkvY2x
vdWQvcmVnaXN0cmF0aW9ucy9kMmI3NTcyZi0zNDA2LTQyYTctYmRjYy04NWYxNzc4NjBkZjciLCJzdWIiOiIvYXBpL3VzZXItcmVnaXN0cmllcy9iMTU4
YjdhYy1kMTEzLTQxNDctODE1OS05ZDliNGViZjE3ZGEvZjdhY2IwNzMtYzU4ZC00ZjhkLWIwZDYtMGNlM2YwYjIyYmNmL3VzZXJzL2ZjMWRlOTE2LWI1N
DAtNDE4MS1iZTJiLWM4ZGU0NGVlYTMzZiIsImlzcyI6IklCTSBBUEkgQ29ubmVjdCIsImV4cCI6MTU5MTcyMzU5NSwiaWF0IjoxNTkxNjk0Nzk1LCJncm
FudF90eXBlIjoicGFzc3dvcmQiLCJ1c2VyX3JlZ2lzdHJ5X3VybCI6Ii9hcGkvdXNlci1yZWdpc3RyYWVzL2IxNThiN2FjLWQxMTMtNDE0Ny04MTU5LTl
kOWI0ZWJmMTdkYS9mN2FjYjA3My1jNThkLTRmOGQtYjBkNi0wY2UzZjBiMjJiY2YiLCJyZWFsbSI6InByb3ZpZGVyL2RlZmF1bHQtaWRwLTIiLCJ1c2Vy
bmFtZSI6InRqIiwiaWRfdG9rZW4iOiJleUpoYkdjaU9pSlNVekkxTmlJc0luUjVjQ0k2SWtwWFZDSXNJbXRwWkNJNklqVXhRamN4T0RCRU9ESkRNME5HU
XpWQlFrSkZPREZHUkRjd05VWTFOekl3TkRBek5FTXpNalFpZlEuZXlKbWFYSnpkRjl1WVcxbElqb2lWR1ZxWVhOMmFXNXBJaXdpYkdGemRGOXVZVzFsSW
pvaVUyRnFhWFFpTENKMWMyVnlYMmxrSWpvaVl6aG1NREF4T1RjdFlqQXhZaTAwWkRnd0xXSXhObU10TURjMk1UUTNaR0pqT1RFMklpd2lkWE5sY201aGJ
XVWlPaUowYWlJc0ltVnRZV2xzSWpvaWRHVnFZWE4yYVc1cExuTmhhbWwwUUdsaWJTNWpiMjBpTENKcFlYUWlPakUxT1RFMk9UUTNPVFY5LlVuRWhRaTFM
TDFzcmlrUE9ReGR0Tkx3SG5iQ24yNnF1U2F5Y2N4WFh0SWstc2YxZEp0T2RaM2k5c2lZdWxkZ0QxV2dueG0wT0UwaVhrRE1kLThsUFVuVzJqc3NaYno4a
k5faXlJVWhmVEZsNmVUNER6WTV6WkJiUmJjYzJZUXlneXQxYnBOOFZnRlJKbm01S2xidUhTXzBkV3BJbVdJeU51em5ndjlhTk9VaVFMVTdCdlpNS0xLQm
t6US1tMGRVSFdzZi1mRlB2cVE5MVUxTy14MVZ4MDBnSHFEQXZlM0VyWExlZm14dGYwalVTY0prcUdjUXVvcE9yNDRNdGcyVUJIRXJhUWpadmlMb0JwUlN
ja25uX1hnRXpZVXdzeUUzQ2d6SC1sRjhJa2Q4VGtqT0tELUc3YmJoRWlZcUhxdnJxVTk0Z18tZ05KbEE0MEZ2b2xnV25GQSIsInNjb3BlcyI6WyJjbG91
ZDp2aWV3IiwiY2xvdWQ6bWFuYWdlIiwicHJvdmlkZXItb3JnOnZpZXciLCJwcm92aWRlci1vcmc6bWFuYWdlIiwib3JnOnZpZXciLCJvcmc6bWFuYWdlI
iwicHJvZHVjdC1kcmFmdHM6dmlldyIsInByb2R1Y3QtZHJhZnRzOmVkaXQiLCJhcGktZHJhZnRzOnZpZXciLCJhcGktZHJhZnRzOmVkaXQiLCJjaGlsZD
p2aWV3IiwiY2hpbGQ6Y3JlYXRlIiwiY2hpbGQ6bWFuYWdlIiwicHJvZHVjdDp2aWV3IiwicHJvZHVjdDpzdGFnZSIsInByb2R1Y3Q6bWFuYWdlIiwiYXB
wcm92YWw6dmlldyIsImFwcHJvdmFsOm1hbmFnZSIsImFwaS1hbmFseXRpY3M6dmlldyIsImFwaS1hbmFseXRpY3M6bWFuYWdlIiwiY29uc3VtZXItb3Jn
OnZpZXciLCJjb25zdW1lci1vcmc6bWFuYWdlIiwiYXBwOnZpZXc6YWxsIiwiYXBwOm1hbmFnZTphbGwiLCJteTp2aWV3IiwibXk6bWFuYWdlIiwid2Via
G9vazp2aWV3Il19.LjfCGN7ymxERSlezc4NhApPj_UDAnSP_N17kDgWwQb_5vjmROu4sDa_VlrbjUMM8jiTv2kqFZdHFtn9c3zgY4aDyqC_AAmhqT3HCV
u_9SQCYK9cVTANxswxBfIRDHArqzJ4MiEw_lSfK8pZnd_XjQ5sg9B-
vKUgJwrB7Bk6X2N6Cm3_QvCQtMCgTevBmwIzE9SIQiOV2041uRj5O0E3NASfacJsWOS4AzSPOMjgxMnWHIgP98MsU0u8v1Hf1VEn1wtrCO7FLiU7DeKYE
a1-1EbqUB0sejwQoKvSYHMFwnh3K8Y97g2lbdjkAcSwVOk02FpUMIIz4aeInYqwlUdzqww",
    "token_type": "Bearer",
    "expires_in": 28800
}
```

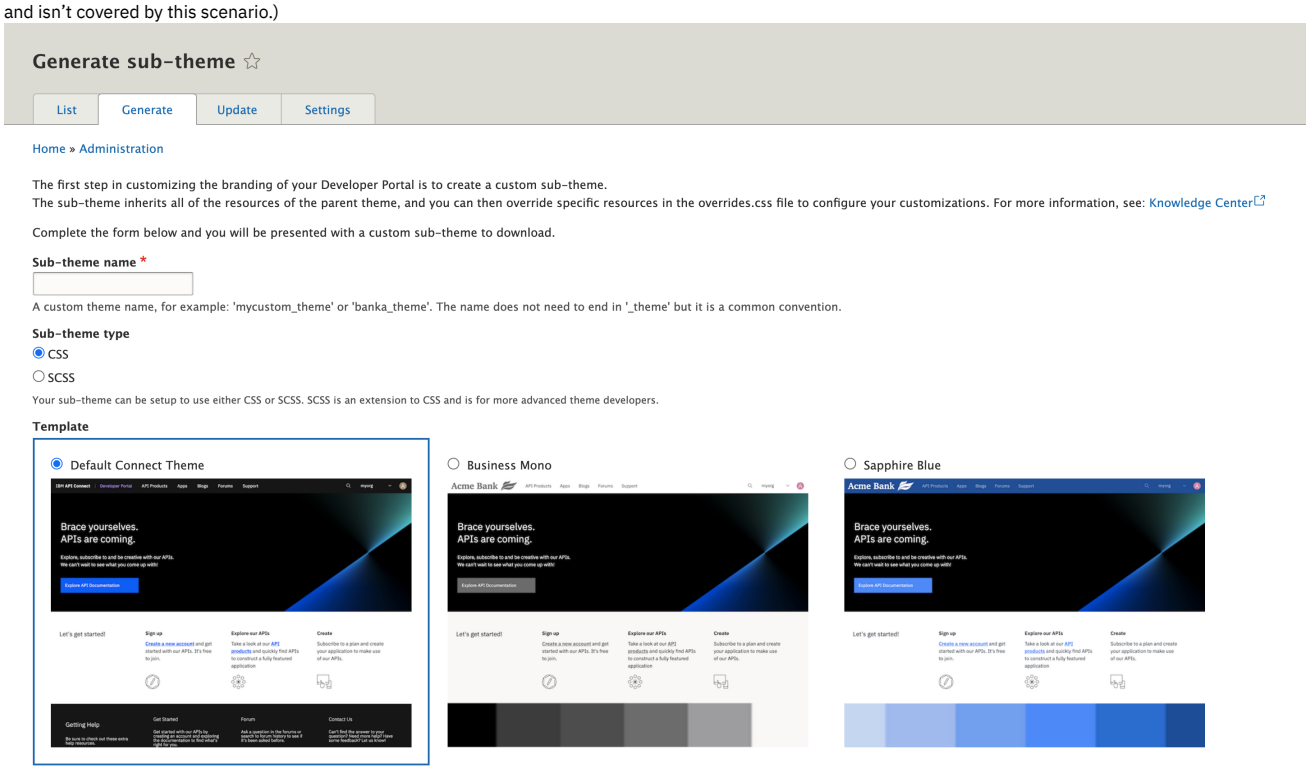

d. Save the token in a variable in your terminal:

```
TOKEN="<paste the access token that you obtained>"
```

You can now authenticate yourself when you invoke the toolkit CLI by using the REST API.

Create a custom theme on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Appearance > Generate sub-theme. The **Generate sub-theme** window is displayed.
4. Enter a Sub-theme name, and select CSS for the Sub-theme type. (If you prefer, you can select SCSS, but this extension to CSS is for advanced theme developers, and isn't covered by this scenario.)

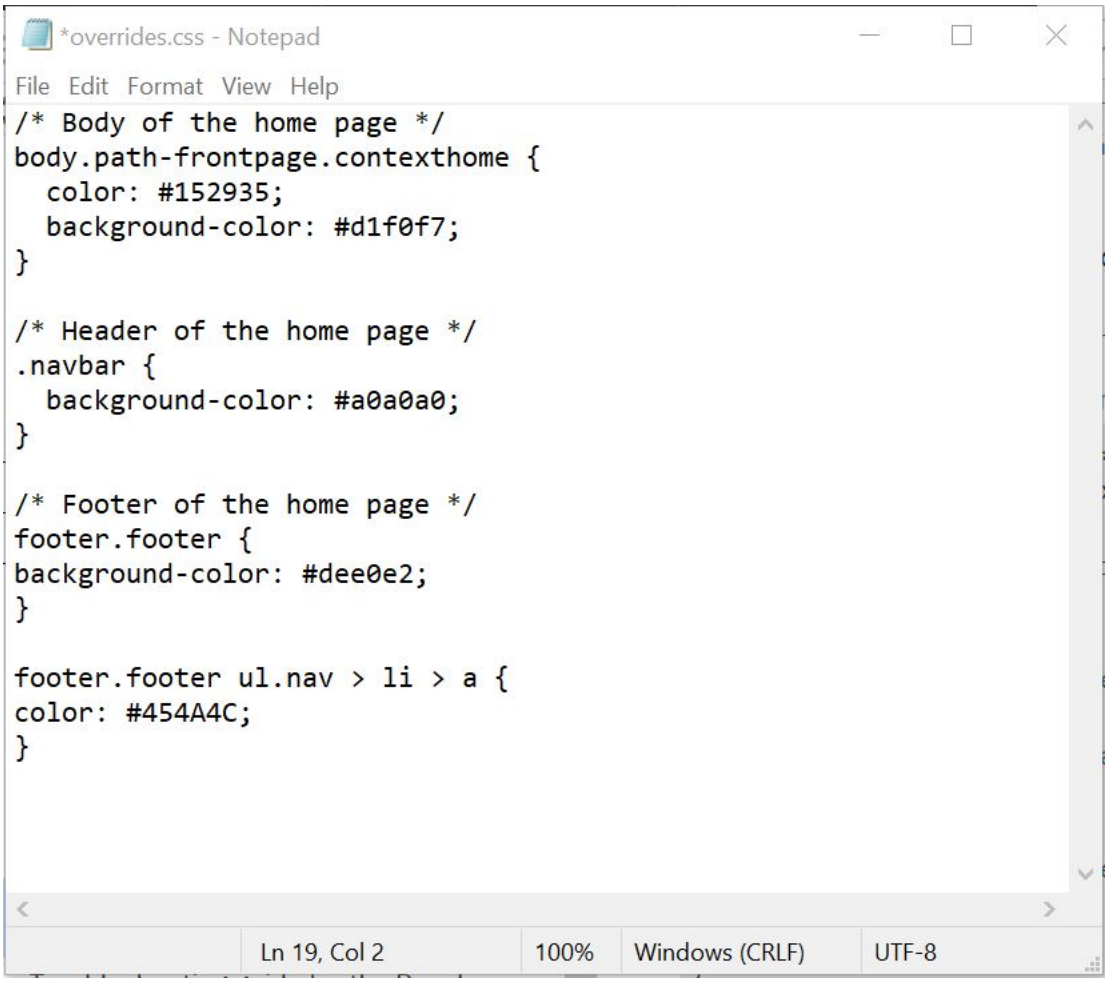


5. Select the Default template to base your **sub-theme** on. You can create a **sub-theme** based on a color template. However, for this scenario you use the default connect_theme template.
6. Click Generate.
7. Download the generated **sub-theme** to a location of your choice, and extract all the files from the .zip file.

On your computer, navigate to the files that you extracted. Find the overrides.css file in here banka_theme/banka_theme/css/overrides.css. You can now customize the overrides.css file.

1. Open the overrides.css file in your chosen editor.
2. Customize your **sub-theme** by entering the following elements into the overrides.css file:

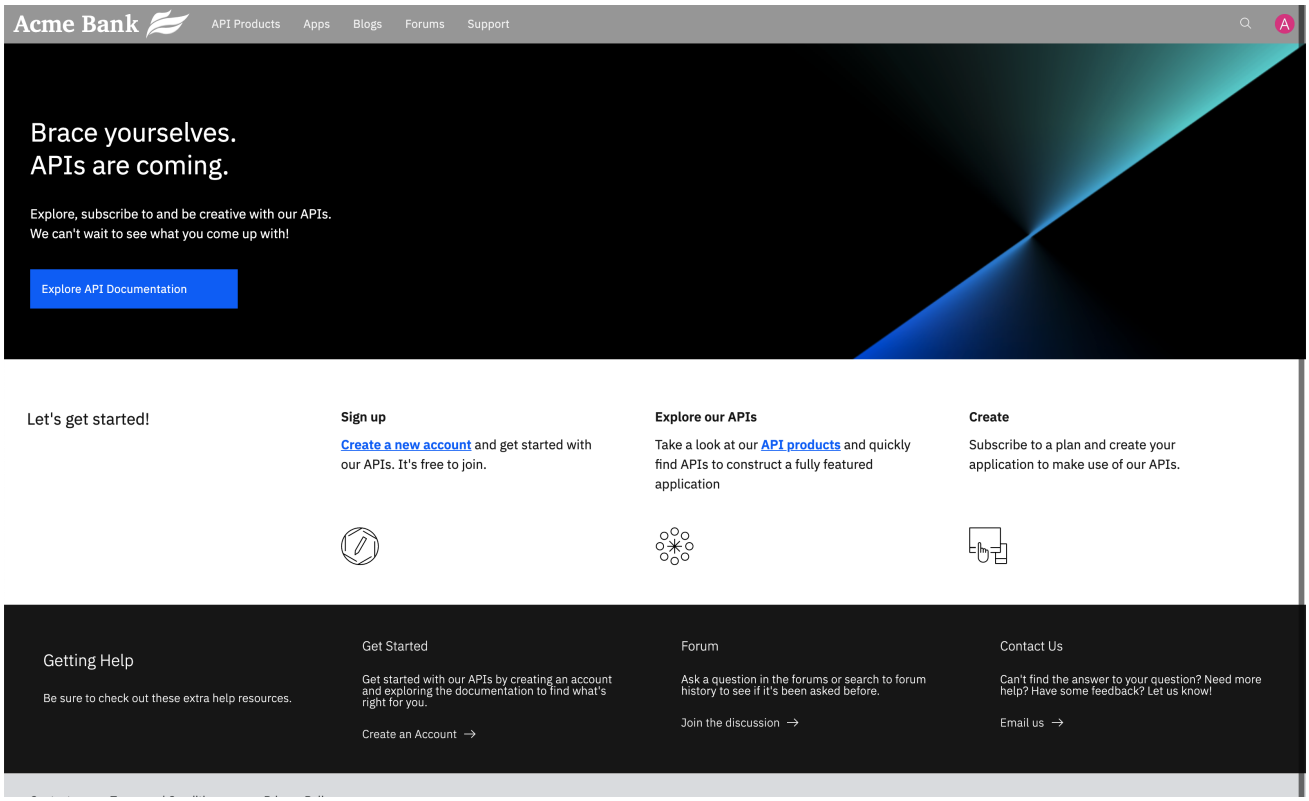
```
/* Body of the home page */
body.path-frontpage.contexthome {
  color: #152935;
  background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
  background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
background-color: #dee0e2;
}

footer.footer ul.nav > li > a {
color: #454A4C;
}
```



3. Save the overrides.css file.

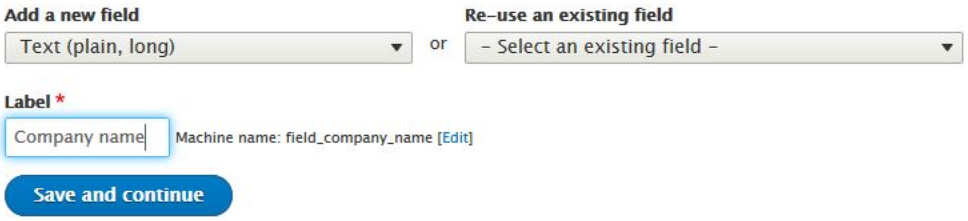
You can now install and enable your customized theme.

1. After you finish updating the overrides.css file, compress all the theme files back into the .zip **sub-theme** file that you downloaded originally.
2. In the Developer Portal, click Appearance > Install new theme. The **Install new theme** window is displayed.
3. In Upload a module or theme archive to install click Browse, and navigate to your newly updated compressed theme file.
4. Click Install to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of **Disabled themes**. Click Enable and set as default to set your new custom **sub-theme** as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



Add a field to an application content type to change the `site-config` on the staging Developer Portal

1. Log in to your staging Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Application > Manage fields.
4. Click + Add field.
5. In **`Add a new field`**, select Text (plain, long). Enter `Company name` as the label title.



6. Click Save and continue.
7. Leave the **`Allowed number of values`** set to 1 and click Save field settings.
8. Click Save field settings. You can now see your new field.

Use the Portal Admin API to export the custom themes and `site-config` on the staging Developer Portal

1. Export your custom themes.
   a. Create the export job by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export'
```

   For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export'
```

   Here is an example of some possible output:

```
status: 201
message:
  stdout: >-
    /catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/f09602a2-a0e4-40dc-a82d-d1af326ad092/custom-
```

```
theme/export/85z2m39znojolv2s/status
  stderr: ''
errors: []
```

b. You can check the status of the export job by invoking the `/status` endpoint.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN"  -H 'Accept: application/yaml'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export/85z2m39znojolv2s/status'
```

Here is an example of some possible output:

```
{
  "status": 200,
  "message": {
    "stdout": "The status of this task is: FINISHED and the artifact can now be streamed"
  },
  "errors": []
}
```

c. You can now download the custom theme file onto your machine.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/export/<task_id>' -o <theme_name>
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/custom-theme/export/85z2m39znojolv2s' -o mytheme.tgz
```

Note: When you use the `curl` commands to download files, if you do not specify an output with `-o`, it incorrectly prints the output as binary onto the terminal screen.

d. On Mac and Linux® machines, you can verify that you have a valid theme export `.tgz` file by invoking the tar `t --list` argument.

```
tar -tvf <theme_name>
```

For example,

```
tar -tvf mytheme.tgz
```

Here is an example of some possible output:

```
drwxrwsr-x  0 aegir  aegir       0  9 Jun 13:40 ./
drwxr-sr-x  0 aegir  aegir       0  9 Jun 13:40 emerald_theme_css/
-rw-r--r--  0 aegir  aegir       7  9 Jun 13:40 emerald_theme_css/emerald_theme_css.theme
-rw-r--r--  0 aegir  aegir   29224  9 Jun 13:40 emerald_theme_css/screenshot.png
-rw-r--r--  0 aegir  aegir     442  9 Jun 13:40 emerald_theme_css/composer.json
-rw-r--r--  0 aegir  aegir     599  9 Jun 13:40 emerald_theme_css/emerald_theme_css.info.yml
-rw-r--r--  0 aegir  aegir    1150  9 Jun 13:40 emerald_theme_css/favicon.ico
-rw-r--r--  0 aegir  aegir    5932  9 Jun 13:40 emerald_theme_css/logo.svg
-rw-r--r--  0 aegir  aegir      61  9 Jun 13:40 emerald_theme_css/emerald_theme_css.libraries.yml
drwxr-sr-x  0 aegir  aegir       0  9 Jun 13:40 emerald_theme_css/css/
-rw-r--r--  0 aegir  aegir   47249  9 Jun 13:40 emerald_theme_css/css/overrides.css
```

2. Export your site config.

a. Create the export job by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export'
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export'
```

Here is an example of some possible output:

```
{
    "status": 201,
    "message": {
        "stdout": "/catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/f09602a2-a0e4-40dc-a82d-
d1af326ad092/config/export/5cy1gk55ciiwn190/status",
        "stderr": ""
    },
    "errors": []
}
```

b. You can check the status of the export job by invoking the `/status` endpoint.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export/5cy1gk55ciiwn190/status'
```

Here is an example of some possible output:

```
{
  "status": 200,
  "message": {
    "stdout": "The status of this task is: FINISHED and the artifact can now be streamed"
  },
  "errors": []
}
```

c. You can now download the site-config file onto your machine.

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/export/<task_id>' -o <config_name>
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/octet-stream'
'https://api.stagingexample.com/portal/catalogs/ibm/staging/config/export/5cy1gk55ciiwn190' -o myconfig.tgz
```

Note: When you use the `curl` commands to download files, if you do not specify an output with `-o`, it incorrectly prints the output as binary onto the terminal screen.

d. On Mac and Linux machines, you can verify that you have a valid theme export `.tgz` file by invoking the tar `t --list` argument. The `head -n 10` option returns the first 10 lines of the result.

```
tar -tvf <config_name> | head -n 10
```

For example,

```
tar -tvf myconfig.tgz | head -n 10
```

Here is an example of some possible output, which should be a list of yaml files:

```
-rw-rw-r--  0 aegir  aegir 393 11 Jun 15:54 block.block.bartik_page_title.yml
-rw-rw-r--  0 aegir  aegir 577 11 Jun 15:54 filter.format.plain_text.yml
-rw-rw-r--  0 aegir  aegir 256 11 Jun 15:54 captcha.captcha_point.consumerorg_invite_user_form.yml
-rw-rw-r--  0 aegir  aegir 741 11 Jun 15:54 block.block.emerald_theme_css_account_menu.yml
-rw-rw-r--  0 aegir  aegir 488 11 Jun 15:54 field.field.node.application.apic_hostname.yml
-rw-rw-r--  0 aegir  aegir 460 11 Jun 15:54 field.storage.node.apic_catalog_id.yml
-rw-rw-r--  0 aegir  aegir 545 11 Jun 15:54 field.field.user.user.apic_user_registry_url.yml
-rw-rw-r--  0 aegir  aegir 427 11 Jun 15:54 language.content_settings.shortcut.default.yml
-rw-rw-r--  0 aegir  aegir 460 11 Jun 15:54 field.storage.node.api_soapversion.yml
-rw-rw-r--  0 aegir  aegir 148 11 Jun 15:54 captcha.captcha_point.node_product_form.yml
```

Use the Portal Admin API to import the custom themes and `site-config` on the production Developer Portal

Note: If you are importing your custom theme onto a new server, then before you continue, follow step 1 of this tutorial to obtain a new token to authenticate with your second server. If you are using the same server, and are importing the theme into a new catalog, then the current TOKEN remains valid for the rest of the steps.
Note: You must import the custom theme before you import the site config. This order is because the site config contains information about the default theme, and so your custom theme must exist on the portal when the site config is imported so that it can be set upon import.

1. Import your custom themes.
   a. On the production catalog, import the custom theme onto the production catalog `prod` by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' -H 'Content-Type:
application/octet-stream' 'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/import' --data-
binary @<path_to_theme_file>
```

For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml' -H 'Content-Type:
application/octet-stream' 'https://api.productionexample.com/portal/catalogs/ibm/prod/custom-theme/import' --data-
binary @/Users/steve/PortaladminCLI/mytheme.tgz
```

Here is an example of some possible output:

```
status: 201
message:
  stdout: >-
    /catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/56f3d4ca-8be2-48f7-8f91-3d654c2b91f2/custom-
theme/import/vtscpucj08i0hfo6/status
  stderr: ''
errors: []
```

   b. You can check the status of the import job by invoking the `/status` endpoint on this command:

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://<management_server>/portal/catalogs/<org>/<catalog>/custom-theme/import/<task_id>/status'
```

For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/yaml'
'https://api.productionexample.com/portal/catalogs/ibm/prod/custom-theme/import/vtscpucj08i0hfo6/status'
```

Here is an example of some possible output:

```
{
  "status": 200,
  "message": {
    "stdout": "2020-06-09 13:00:59: CLI task (custom_theme:import) starting.\n2020-06-09 13:00:59: Importing custom
theme emerald_theme_css\n2020-06-09 13:01:18: CLI task (custom_theme:import) completed successfully.",
    "stderr": ""
  },
```

```
  "errors": []
}
```

2. Import your site config.
   a. Import the existing site-config onto the production catalog `prod` by using this command:

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' -H 'Content-Type:
application/octet-stream' 'https://<management_server>/portal/catalogs/<org>/<catalog>/config/import' --data-binary
@<path_to_config_import>
```

   For example,

```
curl -k -X 'POST' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json' -H 'Content-Type:
application/octet-stream' 'https://api.productionexample.com/portal/catalogs/ibm/prod/config/import' --data-binary
@/Users/steve/PortaladminCLI/myconfig.tgz
```

   Here is an example of some possible output:

```
{
  "status": 201,
  "message": {
      "stdout": "/catalogs/75cfdc05-e818-408e-8ce6-fd3fd61c8756/56f3d4ca-8be2-48f7-8f91-
3d654c2b91f2/config/import/kzyozx4v5sfa8la5/status",
      "stderr": ""
  },
  "errors": []
}
```

   b. You can check the status of the config import job by invoking the `/status` endpoint:

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://<management_server>/portal/catalogs/<org>/<catalog>/config/import/<task_id>/status'
```

   For example,

```
curl -k -X 'GET' -H "Authorization: bearer $TOKEN" -H 'Accept: application/json'
'https://api.productionexample.com/portal/catalogs/ibm/prod/config/import/kzyozx4v5sfa8la5/status'
```

   Here is an example of some possible output:

```
{
  "status": 200,
  "message": {
      "stdout": "2020-06-11 15:42:55: CLI task (site_config:import) starting.\n2020-06-11 15:43:29: CLI task
(site_config:import) completed successfully.",
      "stderr": ""
  },
  "errors": []
}
```

What you did in this scenario

In this scenario, you created a theme and changed the configuration on your staging Developer Portal. Then, you used the Portal Admin API to replicate the changes on to your production Developer Portal.

You can now see the changes that you made on your production Developer Portal.

How to enable and disable maintenance mode on your Developer Portal

You can use the Portal commands in the toolkit CLI to enable and disable maintenance mode on your Developer Portal site.

Before you begin

To complete this Portal CLI scenario, you must have a Developer Portal enabled. You must also have administrator access to this site, and the permission **`settings:manage`**, to complete this scenario. The tutorial Creating the Portal explains how to enable the portal if you haven't enabled it already.

For more information about using the toolkit CLI with the Developer Portal, see Getting started with the Portal CLI commands.

Important: Maintenance mode is designed for short-term site maintenance; it is not meant for long-term usage. While a site is in maintenance mode, the database is not updated with new content from API Manager.

About this scenario

In this scenario, you use the toolkit CLI to enable, and then disable, maintenance mode on your Developer Portal site.

The scenario takes you through the following steps:

1. Log in to the management server as a Provider organization by using the toolkit CLI
2. Use the toolkit CLI to enable maintenance mode
3. Use the toolkit CLI to disable maintenance mode

Log in to the management server as a Provider organization by using the toolkit CLI

- Log in to the toolkit CLI as a Provider org:

```
apic login --server management_server --realm provider/identity_provider --username provider_username --password
provider_password
```

You can determine which identity provider to use in the `--realm` parameter by entering the following command to see a list of all available identity providers (you do not need to be logged in to use this command):

```
apic identity-providers:list --scope provider --server mgmt_endpoint_url --fields title,realm
```

For example,

```
apic identity-providers:list --scope provider --server myserver.com --fields title,realm
total_results: 2
results:
  - title: API Manager User Registry
    realm: provider/default-idp-2
  - title: Corporate LDAP user registry
    realm: provider/corporate-ldap
```

The `title` value should enable you to determine which identity provider to use; you can then copy the corresponding `--realm` parameter directly from the displayed `realm` value. For any identity providers that were created by your administrator after API Connect was installed, the names will have been determined at creation time. The default API Manager Local User Registry for login as a member of a provider organization is `default-idp-2`.
For full details of the `apic login` command, see Logging in to a management server.

Use the toolkit CLI to enable maintenance mode

- Run the following command to enable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --state_key
system.maintenance_mode  --input_format integer --state_value 1
```

Where:
  - *`management_server`* is the endpoint URL of your management server (required).
  - *`catalogid/name`* is the ID or name of the catalog that your site belongs to (required).
  - *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
  - `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
  - `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
  - `--state_value` is the value to assign to the state key (required). Set to `1` to enable maintenance mode.

For example, to enable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key
system.maintenance_mode  --input_format integer --state_value 1
Successfully set state system.maintenance_mode to the value of 1.
```

The Developer Portal site is now inaccessible to everyone apart from an administrator. All other users who enter the site URL will see a maintenance message set by the administrator.

Use the toolkit CLI to disable maintenance mode

- Run the following command to disable maintenance mode on your Developer Portal:

```
apic --mode portaladmin drupal-state:set --server management_server --org orgid/name --catalog catalogid/name --state_key
system.maintenance_mode  --input_format integer --state_value 0
```

Where:
  - *`management_server`* is the endpoint URL of your management server (required).
  - *`catalogid/name`* is the ID or name of the catalog that your site belongs to (required).
  - *`orgid/name`* is the ID or name of the provider organization that the catalog belongs to (required).
  - `--state_key` is the name of the state key (required). Must be set to `system.maintenance_mode`.
  - `--input_format` is the input format of the value of the state key. Must be set to `integer` for configuring the maintenance mode.
  - `--state_value` is the value to assign to the state key (required). Set to `0` to disable maintenance mode.

For example, to disable maintenance mode for a Developer Portal in the `myOrg` organization, and in the `dev` catalog:

```
apic --mode portaladmin drupal-state:set --server my.management.server.com --org myOrg --catalog dev --state_key
system.maintenance_mode  --input_format integer --state_value 0
Successfully set state system.maintenance_mode to the value of 0.
```

Your Developer Portal site is now accessible to everyone again.

What you did in this scenario

In this scenario, you used the toolkit CLI to put your Developer Portal site into maintenance mode, and to turn maintenance mode off again.

Related tasks

- Toggling the site in and out of maintenance mode

Troubleshooting guide for the Developer Portal

Use this guide to help you diagnose and resolve Developer Portal issues in IBM® API Connect. For example, why is the Developer Portal not connecting to the Management server, why is my site not created, and why can't I log in?

About

The following sections provide advice about how to diagnose and resolve some common problems. This information can help you to identify whether a specific IBM API Connect Version 10 component is failing, whether it's an environmental problem, and whether you should raise a support request. Some advice includes reviewing specific log files so you can narrow down the source of the problem.

Note: In the Help page of the Cloud Manager, API Manager, and API Designer user interfaces, there's a Product information tile that you can click to find out information about your product versions, as well as Git information about the package versions being used. Note that the API Designer product information is based on its associated management server, but the Git information is based on where it was downloaded from.

1. Why is the Developer Portal not connecting to the Management server?
2. Why is my Developer Portal site not being created?
3. Why can't I register a user?
4. Why can't I login?
5. Why are changes to Products not appearing in the Developer Portal?
6. Why am I having problems with my Developer Portal user interface?
7. Why am I having problems with my Kubernetes system?
8. Why am I having problems with my Developer Portal backup?
9. Why am I having problems installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal?

Why is the Developer Portal not connecting to the Management server?

Problems with connecting to the Management server can be due to connectivity or TLS issues. You should check the following areas:

- Can the Management server actually reach the Developer Portal?
- Is SSL passthrough enabled or required on any load balancer that sits between the Management server and the Developer Portal? For more information, see JWT security on Kubernetes and OpenShift, JWT security on OVA.

If you need to open a Service Request, include the following information:

- The IBM API Connect MustGather Information.
- Details of the date and time that the connection attempts were made.

Why is my Developer Portal site not being created?

If you are having problems creating a site, you should take the following actions to help diagnose the issue:

1. Tail the nginx container log to confirm that the site create request reached the portal, for example:

   ```
   kubectl logs rb93d4fb118-apic-portal-nginx-7c7b464fb6-vvx5m
   ```

2. Look in the container log for a site create request, for example:

   ```
   POST /portal-service-configuration-create HTTP/1.1" 200
   ```

   a. If you don't find a site create request in the log, then the create request is not reaching the Developer Portal from the Management. Follow the instructions in Why is the Developer Portal not connecting to the Management server? to diagnose the connectivity issues.
   b. If a site create request is in the log, then further diagnostics can be found in the admin pod log, for example:

   ```
   kubectl logs -f rb93d4fb118-apic-portal-www-p4bhj -c admin
   ```

   Look in the admin pod log for messages that refer to `create_site`, as well as the name of the site.

Why can't I register a user?

If you are having problems registering a user, take the following actions to help diagnose the issue:

- The user registration links in the emails are only valid for 24 hours, so check that the system time is correctly configured across all of the pods.
- If emails aren't being received, check that the email server is correctly configured; see Configuring an email server for notifications. Also, if possible, check whether other emails such as publish approval alerts or password reset emails are being received.

If you need to open a Service Request, please take the following steps:

1. Note which user registry type is being used, such as Local User Registry (LUR), or LDAP user registry.
2. Enable the debug REST server and method trace in the Developer Portal UI. Click Configuration > Development > IBM Development on the administrator dashboard. Then select the Enable method entry / exit trace and Enable API Manager REST interface debug check boxes. Click Save configuration.
3. Re-create the issue, making a note of the time, the username, and any messages displayed on the screen.
4. Gather the web container log for the www pod, for example:

   ```
   kubectl logs portal-apic-portal-www-zhqxf web > web.log
   ```

5. Gather all of the Management server logs and the logs for the user registry type that is being used; see IBM API Connect MustGather Information for details.

Why can't I login?

If users are having problems with logging in to the Developer Portal, please take the following steps before opening a Service Request:

1. Note which user registry type is being used, such as Local User Registry (LUR), or LDAP user registry.
2. Enable the debug REST server and method trace in the Developer Portal UI. Click Configuration > Development > IBM Development on the administrator dashboard. Then select the Enable method entry / exit trace and Enable API Manager REST interface debug check boxes. Click Save configuration.
3. Recreate the issue, making a note of the time, the username, and any messages printed to the screen.
4. Gather the web container log for the www pod, for example:

```
kubectl logs portal-apic-portal-www-zhqxf web > web.log
```

5. Gather all of the Management server logs and the logs for the user registry type that is being used; see IBM API Connect MustGather Information for details.

Why are changes to Products not appearing in the Developer Portal?

To determine whether the problem is with the Portal server (either the nginx or the www pod), the Management server, or the network, please make the following diagnostic checks.

Portal nginx pod

1. Tail the nginx pod log on the Portal, for example:

   ```
   kubectl logs -f portal-apic-portal-nginx-866c977c4c-n9gbf
   ```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if the following message appears:

   ```
   ...POST /event-create...
   ```

If the POST message appears in the nginx log, then the publish webhook that was sent from the Management server did arrive at the Portal. If there is an error message in place of a POST message, then the problem is likely to be with the nginx pod. If there is no message at all, then it's likely that the webhook from the Management server is not reaching the Portal and there is a network problem.

Portal www pod

1. Tail the www pod admin container log on the Portal, for example:

   ```
   kubectl logs -f portal-apic-portal-www-dtbzw -c admin
   ```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if the following message appears:

   ```
   Received webhook event 'product_lifecycle' for catalog
   ```

If no such message appears, it suggests that the webhook is either not arriving at the Portal, or not getting beyond the nginx pod (see the previous section Portal nginx pod). Any errors that the admin container has in processing the webhook will be logged in the www pod admin container log.

Management server apim pod

1. Tail the apim pod log on the Management server, for example:

   ```
   kubectl logs -f apiconnect-apim-v2-86b464c4f5-4cx6r
   ```

2. Make a publish request to the Catalog (note that the request must be a Publish or Retire operation, not a Stage operation).
3. In the log being tailed, see if there any error messages.

Search for the word 'sending', you should see the message `webhook: sending successful` if the webhook is sent, or error messages indicating why if the webhook is not sent.

Why am I having problems with my Developer Portal user interface?

If you are having user interface (UI) problems, make the following checks:

- Try clearing the cache from your browser, try using private browsing, and try using different browsers - see whether there are any differences in behavior.
- Check your browser window size and screen resolution, and try increasing and decreasing the settings.
- Investigate the browser developer tools console and network tabs, as they might provide some diagnostic clues.
- Try clearing the Drupal caches from within the Developer Portal; see Clearing the server caches for details.

If you need to open a Service Request, along with the IBM API Connect MustGather Information logs, also include details of the time and date when the problem was reproduced. In addition, an export of the browser developer tools console and network output would be helpful.

Why am I having problems with my Kubernetes system?

Kubernetes provides a DNS server as one of the pods in the kube-system namespace (unless you have configured your Kubernetes setup differently), and sometimes this DNS server can silently fail. When the Portal www or db pods restart, they check that their own hostname and IP address is in the service that is defined for the pod. If the DNS server is down, this check fails. An indication that the DNS server is down is seeing the following log lines printed endlessly in the admin or db containers:

```
[     admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: service-ready:
[     admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: service-all:
[     admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:54: Waiting for sleep in service all check. Pid(s) 352  Seconds
24
[     admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:27:59: Finished waiting for sleep in service all check. Pid(s)
352  Seconds 29
[     admin stdout] 2cd141dc:2cd141dc:2cd141dc 2018-10-08 07:28:01: WARNING 5: r4c09b98d02-apic-portal-admin-all doesn't
include this pod or r4c09b98d02-apic-portal-director does. Checking again in 5 seconds.
```

To fix this problem, delete the DNS pod in the kube-system namespace to force it to restart, for example:

```
kubectl -n kube-system delete pod kube-dns-b76db4f7f-n4lvt
```

Where *kube-dns-b76db4f7f-n4lvt* is the full name of the DNS pod.
Note: To find out the full name of the DNS pod in your Kubernetes system, run the following command:

```
kubectl -n kube-system get pods
```

This command will return data like the following example:

```
$ kubectl -n kube-system get pods
NAME READY STATUS RESTARTS AGE
default-http-backend-5bccfbdc8-mjnnb 1/1 Running 0 4d
etcd-minikube 1/1 Running 0 4d
kube-addon-manager-minikube 1/1 Running 0 4d
kube-apiserver-minikube 1/1 Running 0 4d
kube-controller-manager-minikube 1/1 Running 0 4d
kube-dns-6f4fd4bdf-7zslq 3/3 Running 0 4d
kube-proxy-mddmj 1/1 Running 0 4d
kube-registry-proxy-99qml 1/1 Running 0 4d
kube-registry-v0-zt8pv 1/1 Running 0 4d
kube-scheduler-minikube 1/1 Running 0 4d
kubernetes-dashboard-77d8b98585-dczcs 1/1 Running 0 4d
nginx-ingress-controller-57bcfc76d6-z775d 1/1 Running 0 4d
storage-provisioner 1/1 Running 0 4d
tiller-deploy-587df449fb-bpwhq 1/1 Running 0 4d
```

Why am I having problems with my Developer Portal backup?

If you attempt to back up your portal system or site:

```
apicup subsys exec <portal_subsystem> backup-system|backup-site|backup-all
```

You might encounter the following message:

```
curl: (6) Could not resolve host: apic-portal-apic-portal-director
```

This error is the result of a failure of the SFTP backup file transfer to the backup system, caused by missing authentication with the portal node. To correct this error, `ssh` from the backup system to the portal node, and accept the authentication key when prompted:

1. Enter the portal-www admin container, for example:

   ```
   kubectl exec -it WWW_POD_NAME -c admin bash
   ```

2. `ssh` to your backup server and accept any authentication prompts, for example:

   ```
   ssh BACKUP_SERVER_HOSTNAME
   ```

When you have successfully authenticated with your backup server, you can close the session. This action needs to be done only once. Portal backups can then run without any issues.

Why am I having problems installing Drupal 8 based custom modules or sub-themes into the Drupal 9 based Developer Portal?

From IBM API Connect 10.0.3.0, the Developer Portal is based on the Drupal 9 content management system. If you want to install Drupal 8 custom modules or sub-themes into the Drupal 9 based Developer Portal, you must ensure that they are compatible with Drupal 9, including any custom code that they contain, and not using any deprecated APIs, for example. There are tools available for checking your custom code, such as drupal_check on GitHub, which checks Drupal code for deprecations.

For example, any Developer Portal sites that contain modules or sub-themes that don't contain a Drupal 9 version declaration will fail to upgrade, and errors like the following output will be seen in the `admin` logs:

```
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: emeraldgreen
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for emeraldgreen
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: Checking theme: rubyred
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for rubyred
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: Found themes incompatible with
Drupal 9: emeraldgreen rubyred
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:34:49: check_d9_compat: ERROR: /tmp/restore_site.355ec8 is NOT
Drupal 9 compatible
...
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_1
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '' found for custom_mod_1
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: Checking module: custom_mod_2
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Incompatible
core_version_requirement '8.x' found for custom_mod_2
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: Found modules incompatible with
Drupal 9: emeraldgreen rubyred
[     queue stdout] 14834 729319:355ec8:a7d29c 2021-09-04 20:44:49: check_d9_compat: ERROR: site1.com is NOT Drupal 9
compatible
```

To fix version compatibility errors, all custom modules and sub-themes should declare a `core_version_requirement` key in their *.info.yml file that indicates Drupal 9 compatibility. For example:

```
name: Example module
type: module
description: Purely an example
core: 8.x
core_version_requirement: '^8 || ^9'
package: Example module

# Information added by Drupal.org packaging script on 2020-05-31
version: '8.x-1.3'
project: 'example_module'
datestamp: 1590905415
```

This example specifies that the module is compatible with all versions of Drupal 8 and 9. For more information, see Let Drupal know about your module with an .info.yml file on the drupal.org website.

If you have a backup of a site that you need to restore and are getting the version compatibility error, but the module or theme *.info.yml file cannot be changed easily, then you can modify the site backup.

To modify the site backup, extract the it, edit the relevant files inside it, and then tar the backup file again. Note that this procedure will overwrite the original backup file, so ensure that you keep a separate copy of the original file before you start the extraction. For example:

1. `mkdir /tmp/backup`
2. `cd /tmp/backup`
3. `tar xfz path_to_backup.tar.gz`
4. Edit the custom module and theme files to make them Drupal 9 compatible, and add the correct `core_version_requirement` setting.
5. `rm -f path_to_backup.tar.gz`
6. `tar cfz path_to_backup.tar.gz`
7. `cd /`
8. `rm -rf /tmp/backup`

Related information

- IBM Support Service Requests and PMRs

Migrating your Developer Portal from Version 5 to Version 10

Guidance on how to migrate a Developer Portal site from IBM® API Connect Version 5, to IBM API Connect Version 10.

In IBM API Connect Version 10 the Developer Portal is based on the open source Drupal 9 content management software, which enables more advanced, flexible, and powerful customization capabilities. However, in IBM API Connect Version 5 the Developer Portal is based on Drupal 7. The migration tooling in Version 10 will create a new Developer Portal site, and migrate all of the associated APIs and Products, user information, and subscriptions over. However, customizations such as modules and themes cannot be automatically migrated, as the Drupal 9 baseline is very different to Drupal 7. For more information about upgrading from a Drupal 7 baseline to a Drupal 9 baseline, see Upgrading Drupal in the Drupal documentation.

The following sections take you through the migration process for the Developer Portal, and provide guidance on how to manually migrate any specific customizations to Drupal 9.

- Migration process for the Developer Portal
- Portal Delegated User Registries
- Custom themes
- Custom modules
- Custom Rules
- Custom content types and fields
- Advanced theme development
- Changing the page layout
- Changing the front page
- Features no longer available in the Developer Portal

Note: The ability to migrate is available only from IBM API Connect Version 5.0.8.7 onwards. If your deployment is at an earlier version, you must upgrade before migrating. See Upgrading your API Connect cloud for information about upgrading in Version 5. For more information about the supported versions for migration, see Migrating a Version 5 deployment.
Note: In Version 5, it was possible, though not recommended, to modify files on the Developer Portal server directly. For more information, see Developer Portal best practices for administrators. However, in Version 10, you should not make any system changes. Do not use the **kubectl exec** command to make changes inside the Developer Portal containers, and on OVA deployments do not make any modifications on the Developer Portal VM. Make any customizations on your Developer Portal by using the UI or CLI. Do not run a **kubectl exec** command on any Developer Portal containers unless asked to by IBM.

Migration process for the Developer Portal

The migration process for the Developer Portal migrates all of the following information to Version 10 for each site in your cluster:

- User information (if Portal Delegated User Registry is enabled, the process is slightly different; see Portal Delegated User Registries).
- Products and APIs.
- Consumer organizations.
- Applications.
- Subscriptions.

For information about the main migration process, see Migrating a Version 5 deployment.

Portal Delegated User Registries

In IBM API Connect Version 5, if Portal Delegated User Registry (PDUR) was selected for a Catalog, the user management was delegated from the management server to the Developer Portal. The site administrator within the Portal then performed the configuration of user registries, and the management server recorded that the delegation had taken place, and stored a record of the users that existed in the Catalog. However, the management service maintained no information about the configuration of the specific registries or how users mapped to them. In IBM API Connect Version 10, all user registry configuration takes place in the management server, and PDUR is no longer needed. Therefore, to migrate the PDUR user registries and users from v5 to v10, configuration information must be gathered from both the management server and the Portal node.

If your IBM API Connect Version 5 Developer Portal uses Portal Delegated User Registry (PDUR), you must first export the user information that is stored in the Version 5 Developer Portal local database. This user information can then be fed into the migration tooling so that the correct user registries can be created in the newer version of API Connect.

For more information about how to export the PDUR information from v5, see Exporting Portal Delegated User Registry user information. For more information about the migration process in v10, see Migration steps with PDUR.

Custom themes

If your Version 5 Developer Portal is using a custom theme, you'll need to create the custom theme again for the Drupal 9 baseline. Start by creating a sub-theme of the new Version 10 Developer Portal site by using the theme generator, and then manually customize the sub-theme to your specifications for Drupal 9 by using Cascading Style Sheets (CSS), or Sass CSS (SCSS). Note that using SCSS in your theme makes it far simpler to change colors. We've also added a color template theme generator, which means you can quickly generate a base color theme on which to build your customized site branding. For more information, see Creating a sub-theme.

Note, directly editing the API Connect theme is not permitted or supported, as edited versions of these files are overwritten when a fix pack or iFix is installed.

Other useful resources:

- Tutorial: Creating a custom theme for the Developer Portal - takes you through how to use the theme generator, customize a theme, and install a custom theme.
- Changing the site logo - how to change your site logo.
- Changing the shortcut icon - how to change the shortcut icon (favicon).
- Creating sub-themes - Drupal documentation about creating a sub-themes.
- Theming differences between Drupal 6, 7 & 8 - Drupal documentation about theming differences.

Custom modules

If your Version 5 Developer Portal is using any custom modules, you will need to create these custom modules again for the Drupal 9 baseline. For more information on writing custom modules, see Extend.

Other useful resources:

- Creating custom modules - Drupal documentation about creating custom modules.
- Getting started creating custom modules - Drupal documentation giving background information and prerequisites needed for custom module development.
- Understanding hooks - Drupal documentations about hooks.
- Drupal API hooks - Drupal documentation about the hooks that are available.

Custom Rules

If your Version 5 Developer Portal is using any custom Rules, you will need to create them as custom modules for the Drupal 9 baseline. You cannot export the Drupal 7 Rules, and then import them into the Drupal 9 baseline, as the events and actions are implemented in a different way. For more information about custom modules, see the Custom modules section.

Custom content types and fields

If your Version 5 Developer Portal is using any custom content, you will need to create this content again for the Drupal 9 baseline.

For information about creating custom content types, see the following topics:

- Creating content types
- Adding fields to content types
- Adding custom fields to user records

Advanced theme development

If your Version 5 Developer Portal is using modified content type templates, you must create these templates again for the Drupal 9 baseline. In Drupal 9 the templates are Twig files that define the actual HTML output for a particular piece of content. You can override a Twig template by copying the original template into the templates folder of your custom sub-theme, and then editing the template to your specification. The Developer Portal then uses your template in preference to the original one. For more information, see Applying a modified content type template, and Creating a sub-theme. You can also follow a guided example of how to create a new theme, see Tutorial: Creating a custom theme for the Developer Portal.

Note:
Modifying Twig templates allows very fine-grained control over the structure of pages. However, it also means you might miss new features and defect fixes that are made to the original templates, as your Developer Portal is using your overrides instead of the originals. So, if you override a template, it is your responsibility to check the templates in the latest API Connect Version 10 releases, and to ensure that any equivalent changes that are needed to your overrides to maintain functional equivalence are made.

Changing the page layout

If your Version 5 Developer Portal is using a different page layout, you will need to create this layout again for the Drupal 9 baseline. The page layout is defined by configuring which blocks appear in which regions. Blocks are controlled by using the Structure > Block layout options on the Developer Portal administrator dashboard.

For information about how to configure the page layout, see Adding and changing the blocks displayed on Developer Portal pages.

Changing the front page

If your Version 5 Developer Portal is using a different front page, you will need to create this front page again for the Drupal 9 baseline. The front page is configured from the Structure > Pages options on the Developer Portal administrator dashboard.

For information about how to configure the front page, see Configuring the front page, and Adding custom pages.

Further options that you might want to consider when designing your front page include:

- Change the banner block content or image: Changing the front page banner block.
- Display featured API Products by creating a featured content block: Changing the front page Featured Content block.
- Create custom blocks with your own HTML content: Adding and changing the blocks displayed on Developer Portal pages.
- Set up the social block to include your organization Twitter feed: Integrating X data into the social block.

Features no longer available in the Developer Portal

The following features that are available in IBM API Connect Version 5, are not available in IBM API Connect Version 10:

- Security questions - these can now be created inside an OpenID Connect provider.
- Two-factor authentication - this can now be performed inside an OpenID Connect provider.
- Support tickets - no Drupal 9 equivalent module is currently available.

Guidelines on upgrading your Developer Portal from Drupal 9 to Drupal 10

Information about the upgrade of the Developer Portal from Drupal 9 to Drupal 10.

The Developer Portal is based on the open source Drupal content management system. From API Connect 10.0.5.3, Drupal 9 is upgrading to Drupal 10, which also requires PHP 8.1. All existing Developer Portal sites will be upgraded to, and all new sites will be based on Drupal 10. Therefore, all modules and themes on your Developer Portal site must be compatible with Drupal 10 and PHP 8.1. The core modules and themes are automatically made compatible as part of the upgrade. However, if your Developer Portal sites contain any custom modules or themes, either written in-house or contributed by the Drupal community, it is your responsibility to ensure that they are Drupal 10 and PHP 8.1 compatible before the upgrade.

See the following topics for more detailed information about how to prepare your Developer Portal sites for the upgrade to Drupal 10.

- How to find your custom modules and themes on your Developer Portal.
- Prepare your Developer Portal for the update for Drupal 10.
- Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant.

Upgrading from older versions of API Connect v10

If you are upgrading to 10.0.5.3 from 10.0.1.7 or earlier, or coming from 10.0.5.0 or earlier, your Developer Portal sites will first be upgraded to Drupal 9.5 and then to Drupal 10. This is because upgrading to Drupal 10 requires your site to be on at least Drupal 9.4.4.

Note that if you are in the category where a double upgrade occurs, your Developer Portal sites are going to take double the time to upgrade compared to your normal experience, because upgrading from Drupal 9 to Drupal 10 is a major upgrade. Therefore, when you are planning your maintenance window for upgrading to 10.0.5.3, you need to be aware of the extended time frame. To shorten your upgrade window, consider removing the Developer Portal sites that you do not need.

Further reading

See the following links for additional information about the Developer Portal upgrade to Drupal 10.

- Deprecation checking and correction tools.
- Deprecated modules and themes.
- List of APIs deprecated and now removed from Drupal 10.
- Whats new in PHP 8.1.
- Symfony Upgrade Guide.
- Drush 11 commands.

- **How to find your custom modules and themes on your Developer Portal**
  Guidance on how to find what custom modules and themes you installed on your Developer Portal.
- **Prepare your Developer Portal for the update for Drupal 10**
  Guidance on how to prepare your Developer Portal site from Drupal 9 to Drupal 10.
- **Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant**
  Guidance on how to upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant.

How to find your custom modules and themes on your Developer Portal

Guidance on how to find what custom modules and themes you installed on your Developer Portal.

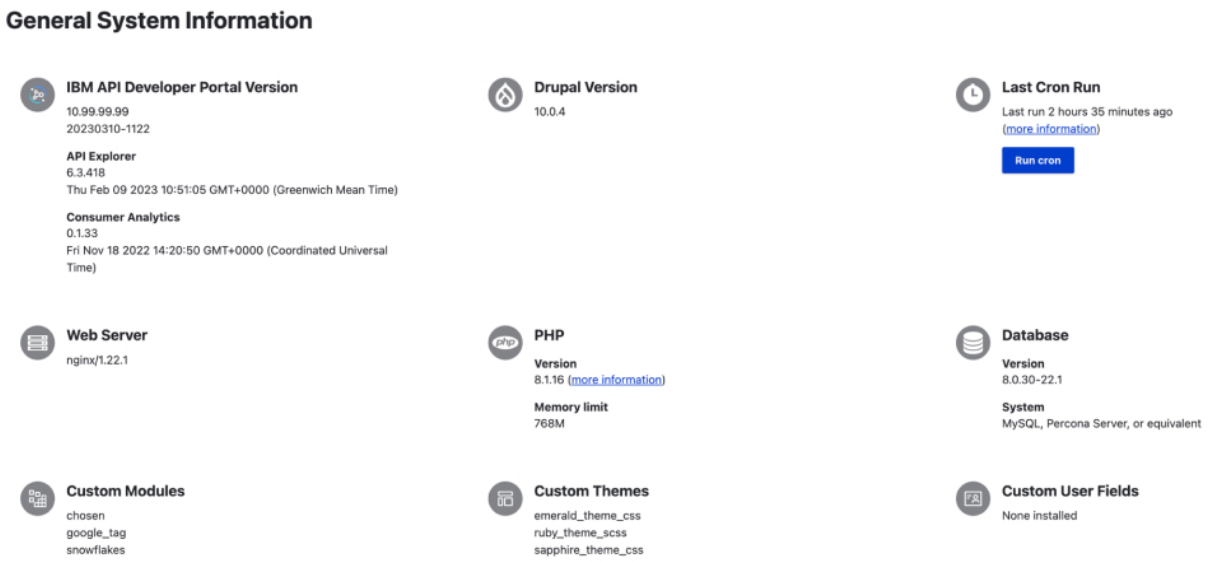
The Developer Portal within API Connect comes with built-in modules and themes that you can use to customize your site. You can extend your customizations by installing external custom modules and themes, or even ones written by yourself. If you are unsure which custom modules and themes are installed on your Developer Portal, you can use one of the following methods to find out.

1. Method 1. Developer Portal admin status report (recommended)
2. Method 2. Developer Portal toolkit CLI
3. Method 3. Developer Portal container

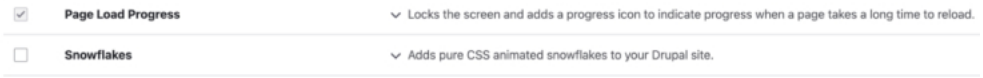
Method 1. Developer Portal admin status report (recommended)

1. Log in to your Developer Portal as the administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Click Manage > Reports > Status report.

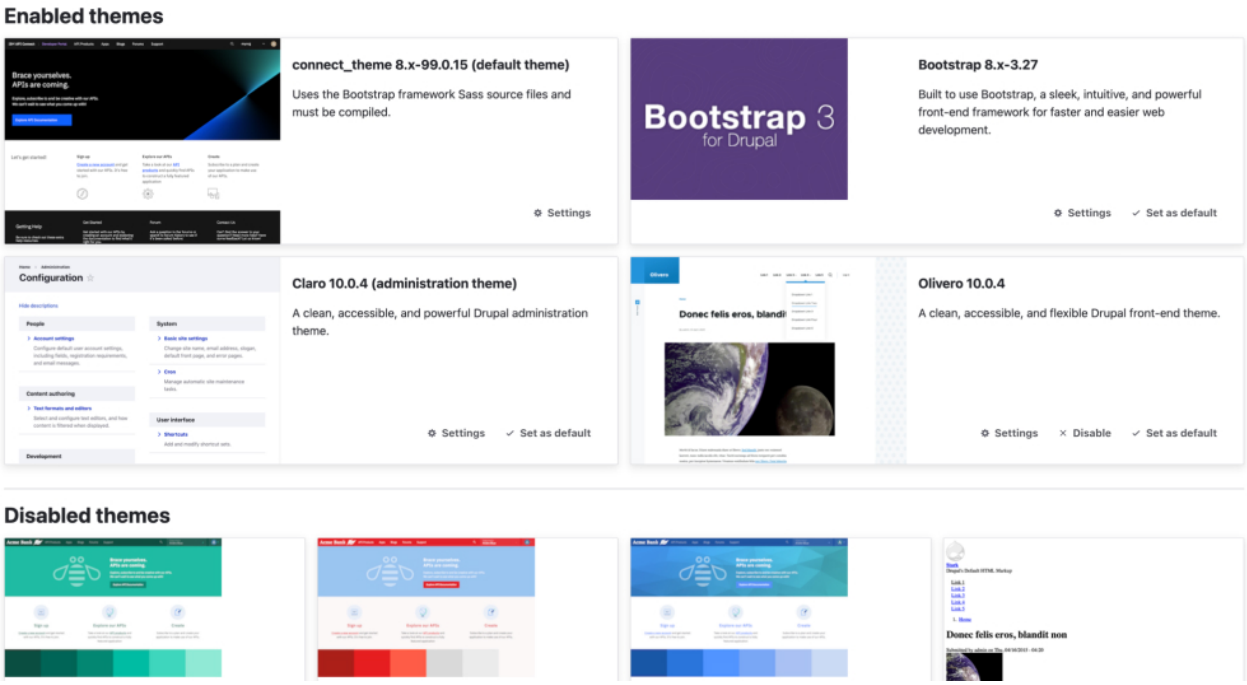
- The custom modules and themes are listed in the Custom Modules and Custom Themes sections at the end of the page.



4. Check which of your custom modules are enabled or disabled by clicking Manage > Extend.
   - The enabled modules are displayed with a selected checkbox, and the disabled ones are not selected as seen in the following example.



5. Check which of your custom themes are enabled or disabled by clicking Manage > Appearance.
   - Find your themes that are categorized into Enabled and Disabled sections.



Method 2. Developer Portal toolkit CLI

Use the **`custom-module:create-export`** and **`custom-theme:create-export`** commands to extract your custom modules and themes into a compressed file.

The compressed file includes a list of your custom modules and themes. For more information about using the Developer Portal toolkit CLI and all the other features it offers, see Overview of the Developer Portal command-line tool.

1. To list your modules, implement the following steps:
   a. Log in to the Developer Portal toolkit CLI as a provider organization.
   b. Run the following **`custom-module:create-export`** command against the Developer Portal.

```
apic custom-module:create-export --mode portaladmin [flags]
```

   c. Specify the appropriate **`org`** and **`catalog`** name.

For example,

```
❯ SERVER=myserver.ibm.com
❯ apic login -s $SERVER -r provider/default-idp-2 -u steve
```

```
Warning: Using default toolkit credentials.
Enter your API Connect credentials
Password?
Logged into myserver.ibm.com successfully

❯ apic -s $SERVER -m portaladmin custom-module:create-export -o ibm -c api-connect-catalog-1
201 CREATED - Task ID: x8jupowmktkf8f2r

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 200:

Message(s) -
The status of this task is: FINISHED and the artifact can now be streamed

Saving File (Large files may take a while)...
Incoming project saved to custom_module_createExport-20230322143819.tgz
```

After the command has run, extract your compressed export to see the modules inside. Or use **tar** directly as shown in the following example.

```
❯ tar --exclude='*/*' -tvf custom_module_createExport-20230322143819.tgz
drwxr-sr-x aegir/aegir       0 2023-03-22 12:05 chosen/
drwxr-sr-x aegir/aegir       0 2023-03-22 12:05 google_tag/
drwxr-sr-x aegir/aegir       0 2023-03-22 12:05 snowflakes/
```

2. To check which one of your custom modules are enabled or disabled, implement the following steps:
   a. Run the following **modules:list** command to print the list of installed modules.

   ```
   apic modules:list --mode portaladmin [flags]
   ```

   b. Search for your custom modules from the result.
   For example,

   ```
   ❯ SERVER=myserver.ibm.com
   ❯ apic -s $SERVER -m portaladmin modules:list -o ibm -c api-connect-catalog-1 --no-core
   .
   .
   .
   chosen:
     package: 'User interface'
     display_name: 'Chosen (chosen)'
     status: Disabled
     version: 3.0.5
   google_tag:
     package: Statistics
     display_name: 'Google Tag Manager (google_tag)'
     status: Enabled
     version: 8.x-1.6
   snowflakes:
     package: 'User interface'
     display_name: 'Snowflakes (snowflakes)'
     status: Enabled
     version: 2.0.1
   .
   .
   ```

3. To list your custom themes, implement the following steps:
   a. Log in to the Developer Portal toolkit CLI as a provider organization.
   b. Run the following **custom-theme:create-export** command against the Developer Portal.

   ```
   apic custom-theme:create-export --mode portaladmin [flags]
   ```

   c. Specify the appropriate **org** and **catalog** name.
   For example,

   ```
   ❯ SERVER=myserver.ibm.com
   ❯ apic login -s $SERVER -r provider/default-idp-2 -u steve
   Warning: Using default toolkit credentials.
   Enter your API Connect credentials
   Password?
   Logged into api.fyre-ci-136264-master.fyre.ibm.com successfully

   ❯ apic -s $SERVER -m portaladmin custom-theme:create-export -o ibm -c api-connect-catalog-1
   201 CREATED - Task ID: oh595vuywxsz3526

   Response Code - 202:

   Message(s) -
   The status of this task is: QUEUED

   Response Code - 200:

   Message(s) -
   The status of this task is: FINISHED and the artifact can now be streamed

   Saving File (Large files may take a while)...
   Incoming project saved to custom_theme_createExport-20230322143856.tgz
   ```

   After the command has run, extract your compressed export to see the themes inside. Or use **tar** to view the contents directly as shown in the following example.

```
❯ tar --exclude='*/*' -tvf custom_theme_createExport-20230322143856.tgz
drwxr-sr-x aegir/aegir       0 2023-03-22 14:36 emerald_theme_css/
drwxr-sr-x aegir/aegir       0 2023-03-22 14:36 ruby_theme_scss/
drwxr-sr-x aegir/aegir       0 2023-03-22 14:36 sapphire_theme_css/
```

4. To check which of your custom themes are enabled or disabled, implement the following steps:
   a. Run the following `themes:list` command to print the list of installed themes.

   ```
   apic themes:list --mode portaladmin [flags]
   ```

   b. Search for your custom themes from the result.
   For example,

   ```
   ❯ SERVER=myserver.ibm.com
   ❯ apic -s $SERVER -m portaladmin themes:list -o ibm -c api-connect-catalog-1 --no-core
   bootstrap:
     package: Bootstrap
     display_name: 'Bootstrap (bootstrap)'
     status: Enabled
     version: 8.x-3.27
   .
   .
   .
   emerald_theme_css:
     package: Bootstrap
     display_name: 'emerald_theme_css (emerald_theme_css)'
     status: Disabled
     version: null
   ruby_theme_scss:
     package: Bootstrap
     display_name: 'ruby_theme_scss (ruby_theme_scss)'
     status: Disabled
     version: null
   sapphire_theme_css:
     package: Bootstrap
     display_name: 'sapphire_theme_css (sapphire_theme_css)'
     status: Disabled
     version: null
   ```

Method 3. Developer Portal container

If you are unable to access the Developer Portal settings or the Developer Portal toolkit CLI, check the Developer Portal container running within your Kubernetes or Red Hat OpenShift environment.

Note: You need access to the Developer Portal Kubernetes pods to be able to perform these steps.

1. Use `kubectl` or `oc` for Red Hat OpenShift environment to obtain a shell to the Developer Portal running container.

   Your Developer Portal sites are located within the `portal-www` pod in the `admin` container within the namespace that your Developer Portal subsystem is deployed to as seen in the following example.

   ```
   ❯ kubectl get pods -n apic | grep www
   portal-014179b4-www-0     2/2     Running     0        8d

   ❯ kubectl exec -n apic -it portal-014179b4-www-0 -c admin -- bash
   bash-4.4$
   ```

2. After you access the container, use the command `list_sites -a` to obtain the site alias for your Developer Portal. For example,

   ```
   bash-4.4$ list_sites -a
   daae4642-b1e2-4fd8-a5a7-77a68ed812ef.944dce4f-bbad-41ce-b166-aa2c9f8c4b17 => ibm.api-connect-catalog-1.portal.fyre-ci-
   136264-master.fyre.ibm.com (INSTALLED)
   daae4642-b1e2-4fd8-a5a7-77a68ed812ef.c560c004-852d-42c3-ae4e-797144a857af => ibm.api-connect-catalog-2.portal.fyre-ci-
   136264-master.fyre.ibm.com (INSTALLED)
   ```

   The Developer Portal `portal.fyre-ci-136264-master.fyre.ibm.com/ibm/api-connect-catalog-1` has the site alias `ibm.api-connect-catalog-1.portal.fyre-ci-136264-master.fyre.ibm.com`.

3. After you identify the Developer Portal that you want to look at, use the alias string within a `find` command to list your modules and themes, as shown in the following example.

   ```
   bash-4.4$ find /web/platforms/devportal*/sites/myserver.ibm.com/modules -mindepth 1 -maxdepth 1 -type d -exec basename {}
   \;
   chosen
   google_tag
   snowflake

   bash-4.4$ find /web/platforms/devportal*/sites/ibm.api-connect-catalog-1.portal.fyre-ci-136264-master.fyre.ibm.com/themes
   -mindepth 1 -maxdepth 1 -type d -exec basename {} \;
   emerald_theme_css
   ruby_theme_scss
   sapphire_theme_css
   ```

4. To check which of your custom modules and themes are enabled or disabled, implement the following sub-steps:
   a. Use the `drush` command-line tool within the container to print the list of installed modules and themes for your Developer Portal.
   b. Then, use the site alias that you obtained earlier to run the following `drush` command and find your custom modules and themes.

   ```
   bash-4.4$ drush @myserver.ibm.com pml --no-core
   Running: drush -y @daae4642-b1e2-4fd8-a5a7-77a68ed812ef-944dce4f-bbad-41ce-b166-aa2c9f8c4b17 pml --no-core
    ------------------------------- ------------------------------------------------------------------------------------
   - ---------- ----------------
     Package                          Name
   ```

```
Status     Version
 ---------------------------------- ----------------------------------------------------------------------------------------------------
- ---------- -----------------
  User                              Account Field Split (account_field_split)
Enabled
  Access control                    ACL (acl)
Enabled
  .
  .
  .
  User interface                    Chosen (chosen)
Disabled   3.0.5
  Statistics                        Google Tag Manager (google_tag)
Disabled   8.x-1.6
  User interface                    Snowflakes (snowflakes)
Disabled   2.0.1

  Bootstrap                         Bootstrap (bootstrap)
Enabled    8.x-3.27
  Bootstrap                         connect_theme (connect_theme)
Enabled    8.x-99.0.15
  Bootstrap                         emerald_theme_css (emerald_theme_css)
Disabled
  Bootstrap                         ruby_theme_scss (ruby_theme_scss)
Disabled
  Bootstrap                         sapphire_theme_css (sapphire_theme_css)
Disabled
```

Results

By using one of the three methods discussed in the topic, you managed to successfully list the custom modules and themes installed on your Developer Portal. This also allowed you to see which of these custom modules and themes are enabled or disabled.

1. Method 1. Developer Portal admin status report (recommended)
2. Method 2. Developer Portal toolkit CLI
3. Method 3. Developer Portal container

Next steps

Ensure that you keep your custom modules and themes up to date, and remove deprecated code within your own written custom modules and themes. If you are not using a custom module or theme, consider removing it. This helps mitigate problems when you want to migrate or upgrade your Developer Portal.

Prepare your Developer Portal for the update for Drupal 10

Guidance on how to prepare your Developer Portal site from Drupal 9 to Drupal 10.

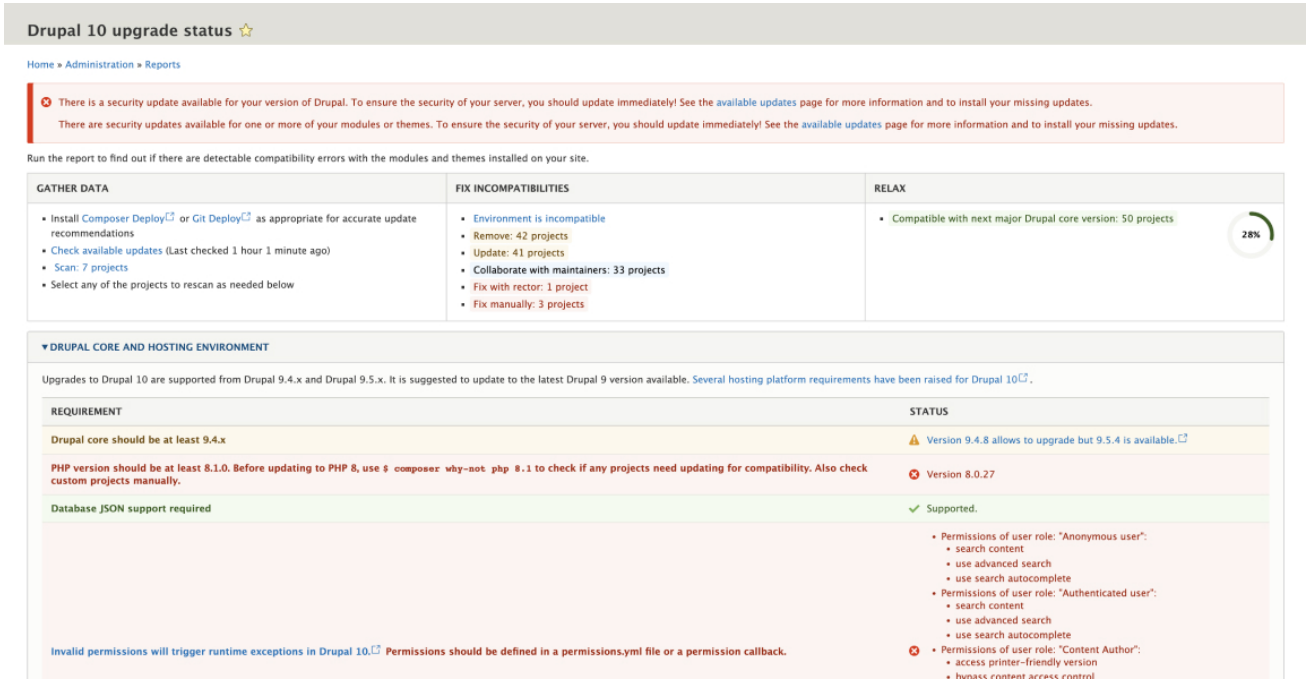
If your Developer Portal contains custom modules and themes that are written by yourself or downloaded from elsewhere, ensure that they are Drupal 10 compatible before you start the upgrade process. This action aims toward a seamless IBM® API Connect upgrade experience. To find more information about your modules and themes compatibility, download the Upgrade Status module available here.

Note: Before you start the upgrade, check the status of your modules and themes and if they are in use. Remove the ones that you do not use.
To learn more on how to prepare your Developer Portal for the Drupal 10 upgrade, check the following steps.

1. Installing the Upgrade Status module
2. Upgrading your module to be Drupal 10 compatible
3. Import your module into your Developer Portal
4. Verifying that your module is Drupal 10 compatible

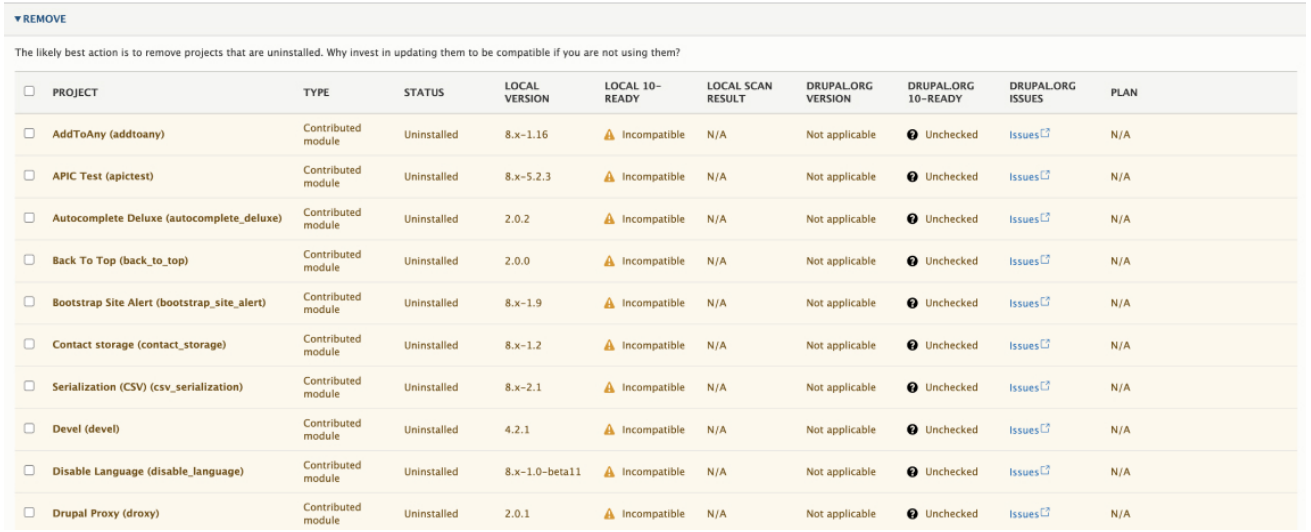
Installing the Upgrade Status module

1. Log in to your Developer Portal as the administrator.
2. If the administrator dashboard is not displayed, click Manage to display it.
3. Install the Upgrade Status module. For more information about how to install the Upgrade Status, see Installing custom modules.
4. When you finish your Upgrade Status installment, click Admin > Reports > Upgrade-status.

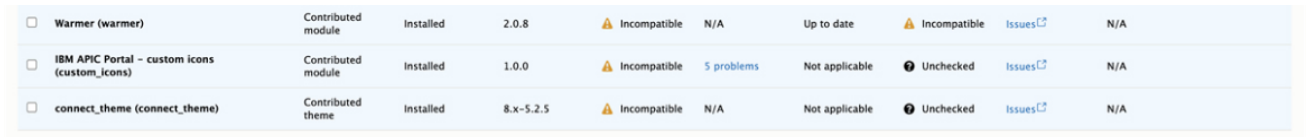


5. The DRUPAL CORE AND HOSTING ENVIRONMENT section can be ignored. This area is managed by IBM API Connect and gets resolved when you upgrade.
   Note: Your custom modules and themes must be PHP 8.1 compatible before you upgrade to Drupal 10, and you are responsible for doing the upgrade.
6. The list of projects further on the page are the ones that your Upgrade Status believes are not compatible with Drupal 10. These projects are categorized into sections, for example: REMOVE, UPDATE, COLLABORATE WITH MAINTAINERS.

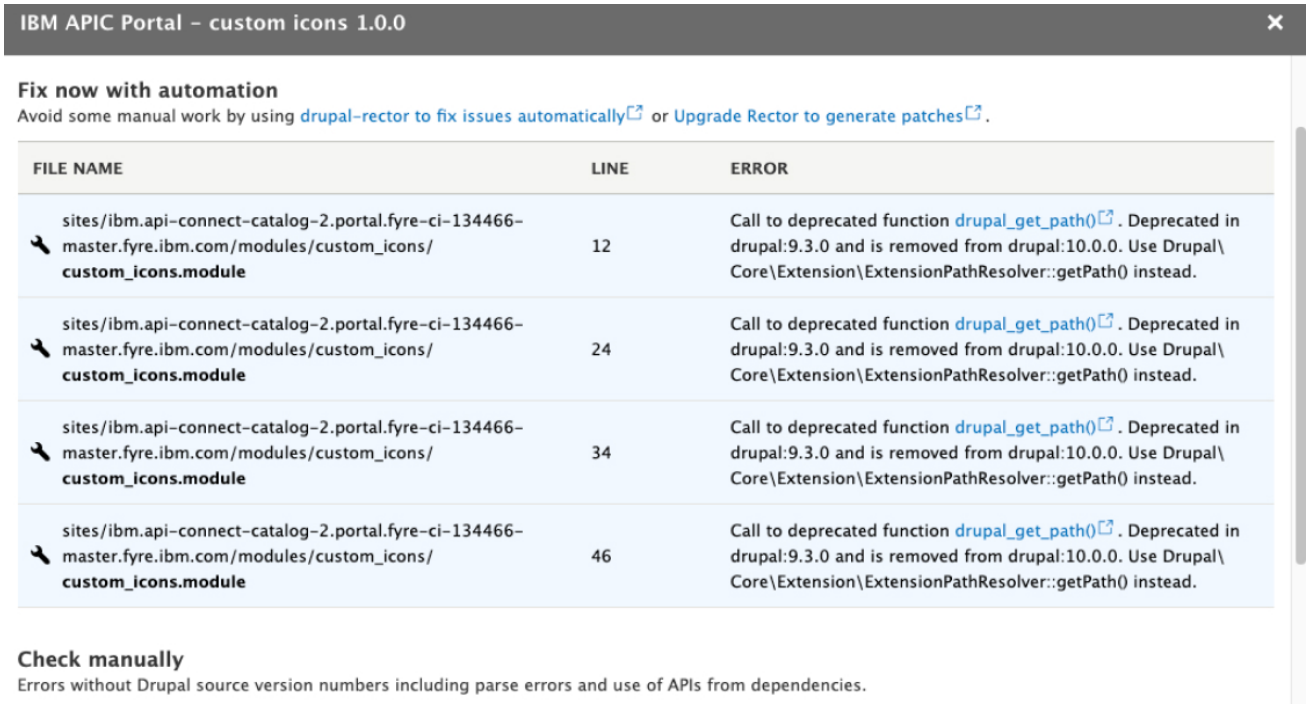
   When you upgrade IBM API Connect, the Drupal 10 compatibility is automatically given to the modules and themes that you have not created or installed manually.



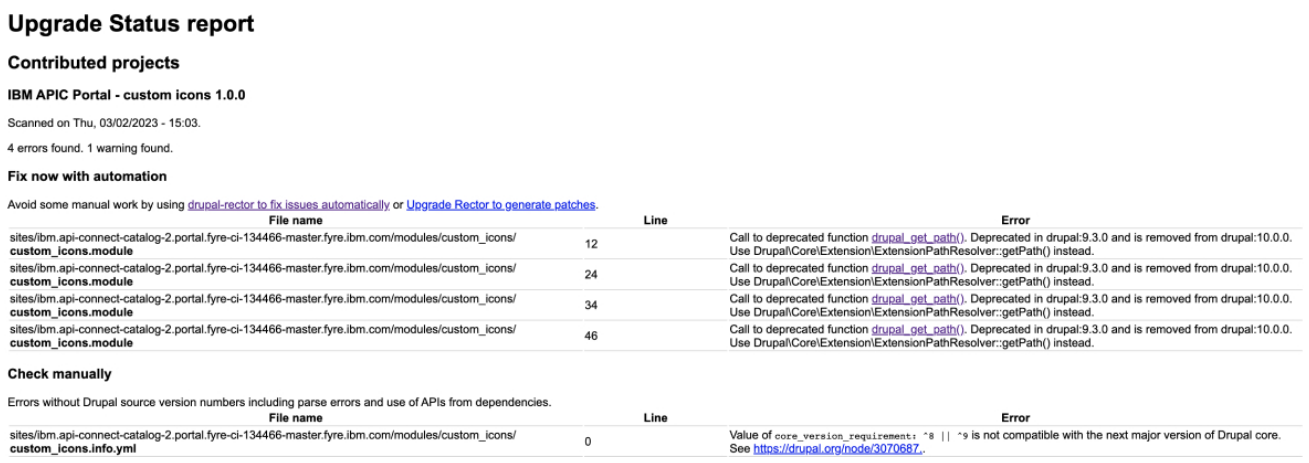
7. If you find your custom modules and themes there, you can scan them to reveal what changes are needed to make them Drupal 10 compatible. For instance, the following image demonstrates the process of scanning an installed custom module, named **`custom_icons`**.



8. Select your custom modules and themes and then click Scan selected. When completed, if your custom module or theme requires changes, you see a link on the row of them.
   Note: For Upgrade Status to identify all of the Drupal 10 compatibility issues in your custom modules and themes, you must have them enabled before scanning it.



9. Click export as HTML to see a clear view.



Upgrade Status report

Contributed projects

IBM APIC Portal - custom icons 1.0.0

Scanned on Thu, 03/02/2023 - 15:03.

4 errors found. 1 warning found.

Fix now with automation

Avoid some manual work by using drupal-rector to fix issues automatically or Upgrade Rector to generate patches.

File name	Line	Error
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	12	Call to deprecated function drupal_get_path(). Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	24	Call to deprecated function drupal_get_path(). Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	34	Call to deprecated function drupal_get_path(). Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.module	46	Call to deprecated function drupal_get_path(). Deprecated in drupal:9.3.0 and is removed from drupal:10.0.0. Use Drupal\Core\Extension\ExtensionPathResolver::getPath() instead.

Check manually

Errors without Drupal source version numbers including parse errors and use of APIs from dependencies.

File name	Line	Error
sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons/custom_icons.info.yml	0	Value of core_version_requirement: ^8 || ^9 is not compatible with the next major version of Drupal core. See https://drupal.org/node/3070687.

Upgrading your module to be Drupal 10 compatible

This module is an example of how you can read the report and fix your code to be compatible with Drupal 10.

1. The report shows that drupal_get_path() is deprecated, and the code must be changed to use the newer function **`getPath()`**. The following examples demonstrate the change to the newer extension type-specific version.

Before

```
 $name = preg_replace("/[^A-Za-z0-9]/", '', $returnValue);
  $returnValue = Url::fromUri('internal:/' . drupal_get_path('module', 'custom_icons') . '/images/' .
_custom_image_getRandomImageName($name))
    ->toString();
```

After

```
$name = preg_replace("/[^A-Za-z0-9]/", '', $returnValue);
  $iconPath = \Drupal::service('extension.list.module')->getPath('custom_icons');
  $returnValue = Url::fromUri('internal:/' . $iconPath . '/images/' . _custom_image_getRandomImageName($name))
    ->toString();
```

2. Add the new version into the **`info.yml`** file.
3. **`Drupal/Core/Extension/Extension List`** is still present in Drupal 8 and Drupal 9, so you can keep them, and add on support for Drupal 10.
4. Modify the version number when you make a new change to the module as seen in the following example.

Before

```
 name: 'IBM APIC Portal - custom icons'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of custom product icons'
package: 'Custom'
core_version_requirement: ^8 || ^9
```

```
version: 1.0.1
project: 'custom_icons'
dependencies:
  - ibm_apim

After

name: 'IBM APIC Portal - custom icons'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of custom product icons'
package: 'Custom'
core_version_requirement: ^8 || ^9 || ^10
version: 1.0.2
project: 'custom_icons'
dependencies:
  - ibm_apim
```

Import your module into your Developer Portal

Import the module by using the Developer Portal Toolkit CLI. For more information about how to use the Developer Portal CLI, see Getting started with the Portal CLI commands.

For example,

```
❯ apic login -s $SERVER -r provider/default-idp-2 -u steve
Warning: Using default toolkit credentials.
Enter your API Connect credentials
Password?
Logged into api.fyre-ci-134466-master.fyre.ibm.com successfully

❯ apic -s $SERVER -m portaladmin custom-module:create-import -o ibm -c api-connect-catalog-2 /tmp/custom_icons.zip
Loading File (Large files may take a while)...

201 CREATED - Task ID: wp5fhmm9l74pmkw1

Response Code - 202:

Message(s) -
The status of this task is: QUEUED

Response Code - 202:

Message(s) -
The status of this task is: RUNNING

Response Code - 200:

Message(s) -
2023-02-27 14:39:38: CLI task (custom_module:import) starting.
2023-02-27 14:39:40: Deleting existing custom module /var/aegir/platforms/devportal-9.x-10.0.5.2-20230105-1720/sites/ibm.api-connect-catalog-2.portal.fyre-ci-134466-master.fyre.ibm.com/modules/custom_icons
2023-02-27 14:39:40: Importing custom module custom_icons
2023-02-27 14:40:25: CLI task (custom_module:import) completed successfully.
```

Alternatively, import your updated module by installing it again using the Developer Portal UI. For more information about how to install custom modules, see Installing custom modules.

Verifying that your module is Drupal 10 compatible

The updated module is now imported into the Developer Portal. Return to the Upgrade Status report page to rescan the module. When it becomes compatible with Drupal 10, it appears at the end of the page.

Workbench (workbench)	Contributed module	Installed	8.x-1.4	✓ Compatible	N/A	Up to date	✓ Compatible	Issues	N/A
IBM APIC Portal - custom icons (custom_icons)	Contributed module	Installed	1.0.0	✓ Compatible	No problems found	Not applicable	? Unchecked	Issues	N/A
Snowflakes (snowflakes)	Contributed module	Uninstalled	2.0.1	✓ Compatible	3 problems	Not applicable	? Unchecked	Issues	N/A

Results

You used the Upgrade Status module to identify and fix the Drupal 10 compatibility problems with your custom modules and themes. However, this process might not cover everything that you need to be ready for the upgrade to Drupal 10. Drupal 10 requires PHP 8.1, so you might need to change your custom modules and themes to support it. Remember to keep your custom modules and themes up to date, and to remove deprecated code within your own written custom modules and themes.

Upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant

Guidance on how to upgrade your custom modules to be Drupal 10 and PHP 8.1 compliant.

The following sections provide examples of the most common updates that you might need to make. However, you must check the deprecation lists for Drupal and PHP to ensure that you cover all of the scenarios in your Developer Portal sites.

Notable changes the update brings

The following items are some of the key changes in Drupal 10.

- **`CKEditor4`** is removed and **`CKEditor5`** is now the default text editor.
- **`Claro`** is now the default theme for the Admin UI, which gives it a modern and refreshing look.
- Ensure that none of your custom modules depend on them as they will not be there after the upgrade.

  The following modules are uninstalled as they do not support Drupal 10:
  - ckeditor_media_embed
  - console
  - flood_unblock
  - message
  - message_notify
  - message_subscribe
  - video_embed_field
  - video_embed_html5
  - content_browser
  - entity_browser
  - entity_browser_enhanced
  - entity_embed
  - file_browser
  - editor_file
  - color
  - Unlimited_number
  - embed
  - ckeditor
  - quickedit
- The Drupal 10 upgrade comes with a major Symfony upgrade version. Drupal 9 used Symfony 4.4, and Drupal 10 now uses Symfony 6, so if you are using any Symfony ensure that they have not been deprecated. Symfony provides a guide at the following link https://symfony.com/doc/current/setup/upgrade_major.html.
- The default Drush version is now Drush 11. If you are writing Drush commands in your custom modules, you must ensure they are Drush 11 compatible. The following links provide examples on how you can achieve compatibility, https://www.specbee.com/blogs/writing-your-own-custom-drush-9-and-10-commands and https://www.drush.org/11.x/commands/.

PHP 8.1 deprecations

The following items are some of the key PHP 8.1 deprecations.

Passing null to non-nullable internal function parameters is deprecated
: There are multiple scenarios that require a fix with the possibility to pass a **`null`** to some PHP functions.
The following example is the most common PHP deprecation.
`Before:`

```
$redacted_url = $url;
if (strlen($redacted_url) > 0 && strpos($redacted_url, '/') !== 0) {
```

`After:`

```
$redacted_url = $url ?? '';
if (strlen($redacted_url) > 0 && strpos($redacted_url, '/') !== 0) {
```

Optional parameters specified before the required parameters
: Some functions in code have optional parameters before the required ones. However, in PHP 8.1, the required parameters come after the optional ones.

Improved type safety
: PHP 8.1 improves the type of safety, which can result in numerous warnings that might require changing functions and variable declarations.
The following example addresses the improved safety and the optional parameters before required parameters deprecations.
`Before:`

```
protected $id;

public function oldFunction($url = '', $data) {
  return $id;
}
```

`After:`

```

/**
 * Drupal entity ID
```

```
 *
 * @var string
 */
protected $id;

public function newFunction(someObject $data, string $url = ''): string {
  return $id;
}
```

Drupal 10 deprecations

The following items are some of the key PHP 8.1 deprecations.

Entity queries now require an `accessCheck`

If you have custom modules with entities, you must do an `accessCheck`. For more information about access checking, see https://www.drupal.org/node/3201242.
For example,
Before:

```
$query = \Drupal::entityQuery('node');
$query->condition('type', 'api');
$query->condition('status', 1);
$nids = $query->execute();
```

After:

```
$query = \Drupal::entityQuery('node');
$query->condition('type', 'api');
$query->condition('status', 1);
$nids = $query->accessCheck()->execute();
```

The `Once` function is removed from the jQuery library and added to its own core library

If you have any modules that use the `once`function in your JavaScript files, you need an update to pull in the new library. For more information about how to remove jQuery dependency, see https://www.drupal.org/node/3158256.
For more examples on how you might upgrade your code, see https://www.drupal.org/project/drupal/issues/3183149.
For example,
Before:

```
// ibm_apim.libraries.yml
validate_password:
  version: 1.x

// validate_password.js
(function($, Drupal, drupalSettings) {
  var $passwordInput = $(context).find('input.js-password-field').once('ibmApimValidatePassword');
})(jQuery, Drupal, drupalSettings);
```

After:

```
// ibm_apim.libraries.yml
validate_password:
  version: 1.x
  dependencies:
    - core/once

// validate_password.js
(function($, Drupal, drupalSettings, once) {
  var $passwordInput = $(once('ibmApimValidatePassword', context)).find('input.js-password-field');
})(jQuery, Drupal, drupalSettings, once);
```

The `app.root` and `site.path` services that are converted into container parameters

If you use the `app.root` and `site.path` services in your custom modules, you must ensure they are converted into container parameters.
For example,
Before:

```
 \Drupal::service('config.factory')
    ->getEditable('locale.settings')
    ->set('translation.use_source', 'local')
    ->set('translation.path', \Drupal::service('site.path') . '/translations')
    ->save();
```

After:

```
\Drupal::service('config.factory')
    ->getEditable('locale.settings')
    ->set('translation.use_source', 'local')
    ->set('translation.path', \Drupal::getContainer()->getParameter('site.path') . '/translations')
    ->save();
```

The parameters of some `Forms`are updated, so be aware if you are extending them.

If you are extending your `Forms`, note the parameters of some of them have changed. For more information about the updated parameters, see https://www.drupal.org/node/3251987.

Next steps

This topic covers the key upgrades you might need when upgrading your custom modules to be Drupal 10 and PHP 8.1 compliant. However, note that it is your responsibility to check the deprecation lists for Drupal and PHP to ensure that you cover all of the scenarios in your Developer Portal sites.

For more information about making your custom modules Drupal 10 and PHP 8.1 compliant, see the following links:

- Deprecation checking and correction tools.
- Deprecated modules and themes.
- List of APIs deprecated and now removed from Drupal 10.
- Whats new in PHP 8.1.
- Symfony Upgrade Guide.
- Drush 11 commands.

Developer Portal tutorials

Tutorials for using the Developer Portal.

The following guided examples help you through some of the most common user scenarios in the Developer Portal, as well as some of the more complex areas.

Developer Portal tutorials by category
For API consumers Create an Application
User authentication Adding a field to the sign up form Adding validation to a field on the sign up form Adding a field group to group fields in a content type
Getting started configuring the Developer Portal Creating the Portal Creating a private Portal site Customizing themes
Advanced customization Changing the front page Grouping products by category Adding a custom block to the front page Adding a custom block to a page other than the front page Using avatars Using image optimization Adding a Back to top button Creating a drop-down menu link Displaying blog posts on the front page Displaying tweets on the front page Changing the profile page to display firstname lastname, instead of username Using a custom weighting sort order on the product list page Configuring the RobotsTxt file Configuring the SecurityTxt file Synchronizing application credentials with an external server

Tutorial: Creating the Developer Portal

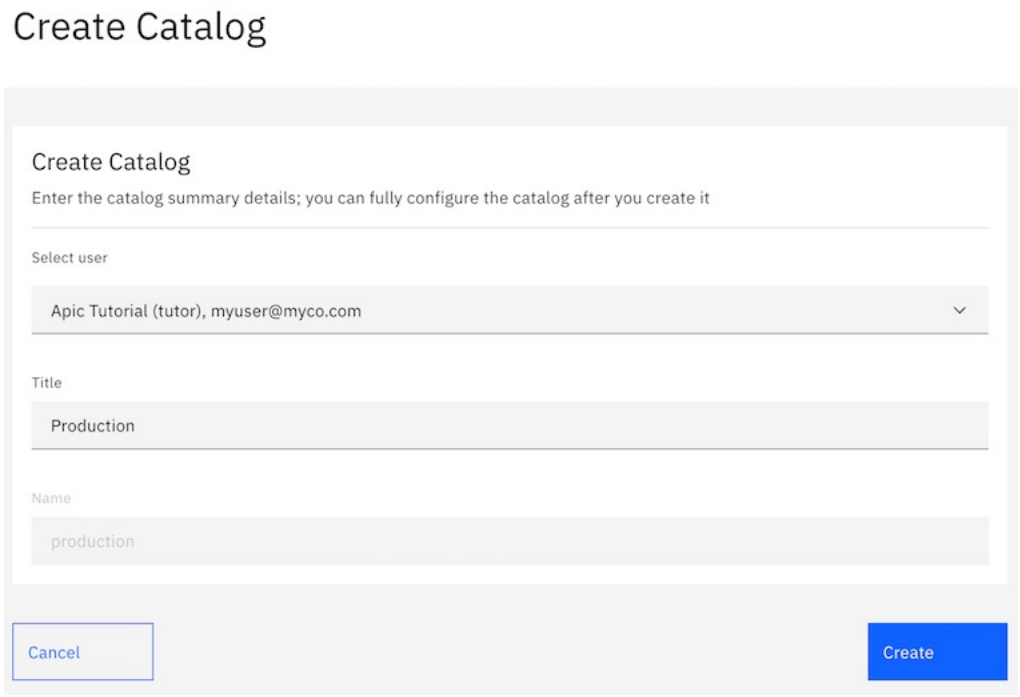
About this task

In this tutorial, you are going to complete the following lessons:

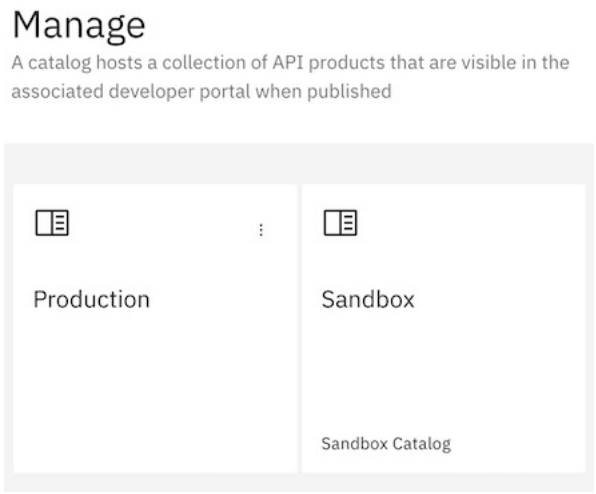
Enabling the Developer Portal through the API Manager

By proceeding with the following steps, you create an administrator account:

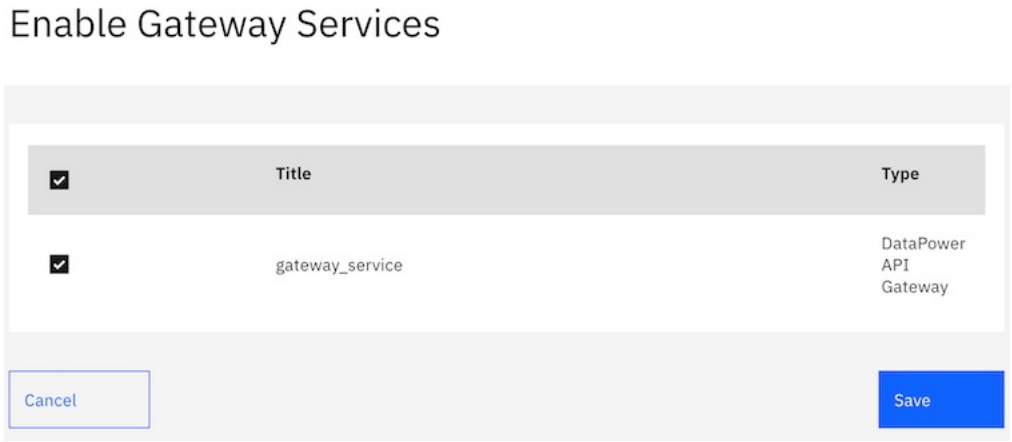
1. Log in to the API Manager
2. Click Manage, then click Add, and select Create catalog.
3. Select the owner from the **`Select user`** list, then enter a **`Title`** for your catalog, for example `Production`.



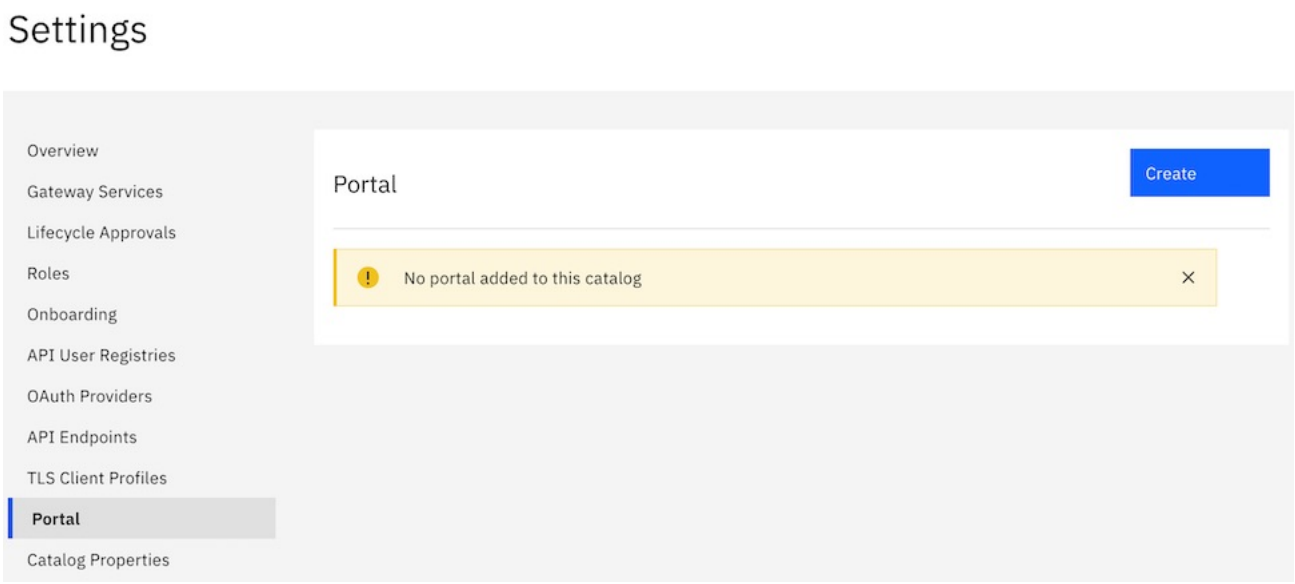
4. Click Create. The Manage page appears.



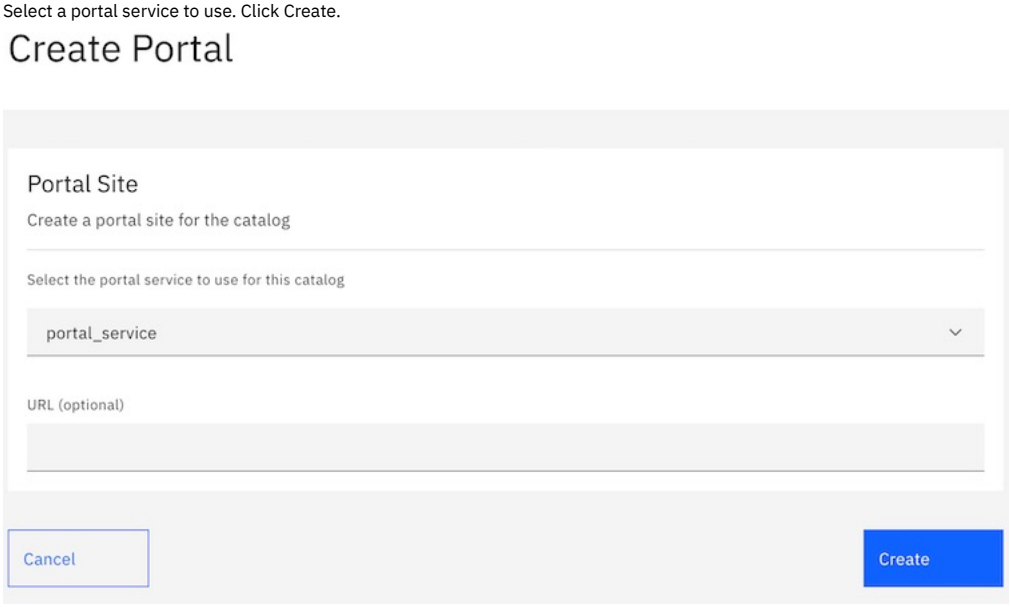
5. Click on the tile that corresponds to your new catalog, for example **`Production`**.
6. Click Catalog settings.
7. Click Gateway Services. If the message **`You currently don't have any gateway services`** is displayed, click Edit, select the configured gateway service, and click Save.



8. Click Portal.
9. Click Create.



10. Select a portal service to use. Click Create.



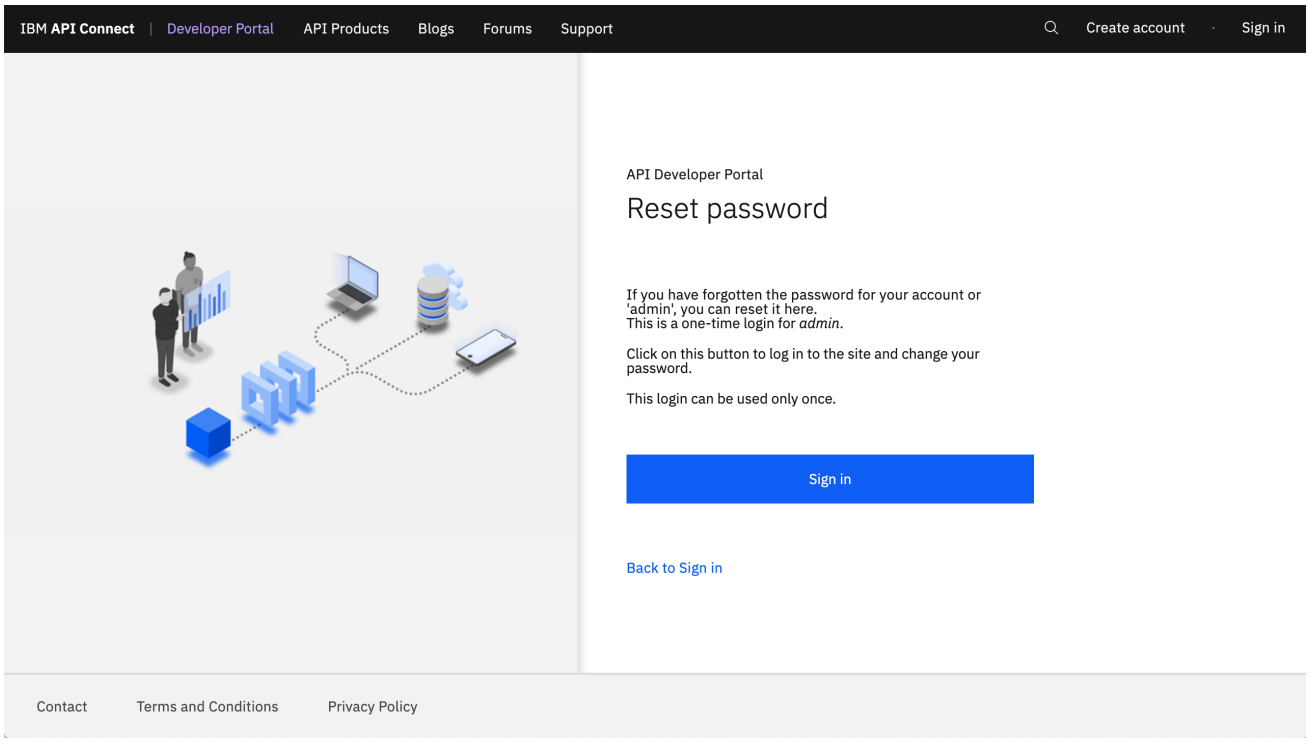
After a few minutes, you receive an email with a link to your Developer Portal site for that catalog. The link is a single use only link for the administrator account.

Administrator (admin),

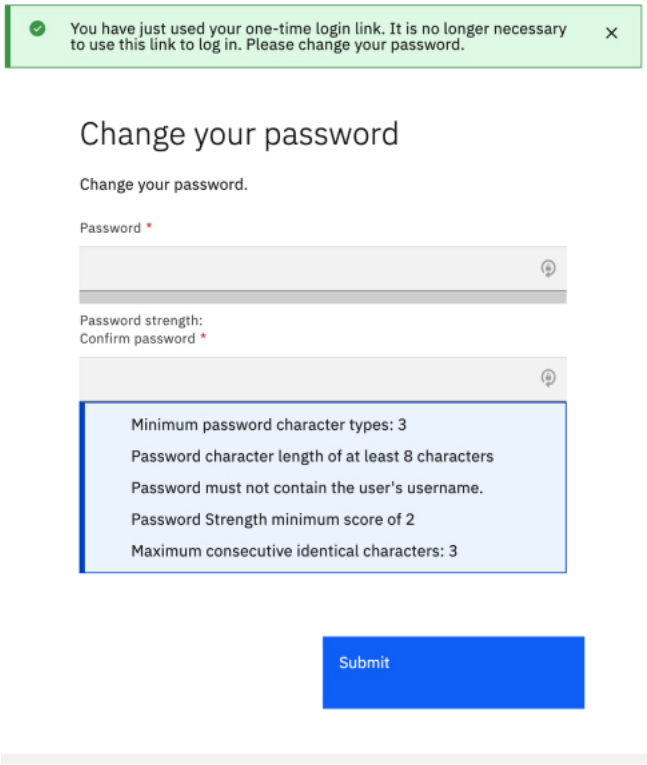
Your Developer Portal site has been created. You can log in as the 'admin' user by using the following one-time log in link. After you have logged in with the following link, you must change the password for the 'admin' user account immediately.

https://portal.apidemo.demo.ibm.com/tutorial/production/user/reset/1/1591727237/h26vxzf2e51UqqK7vf10LNbHiVEC5VQ_AZAoKCYH72C/new/login

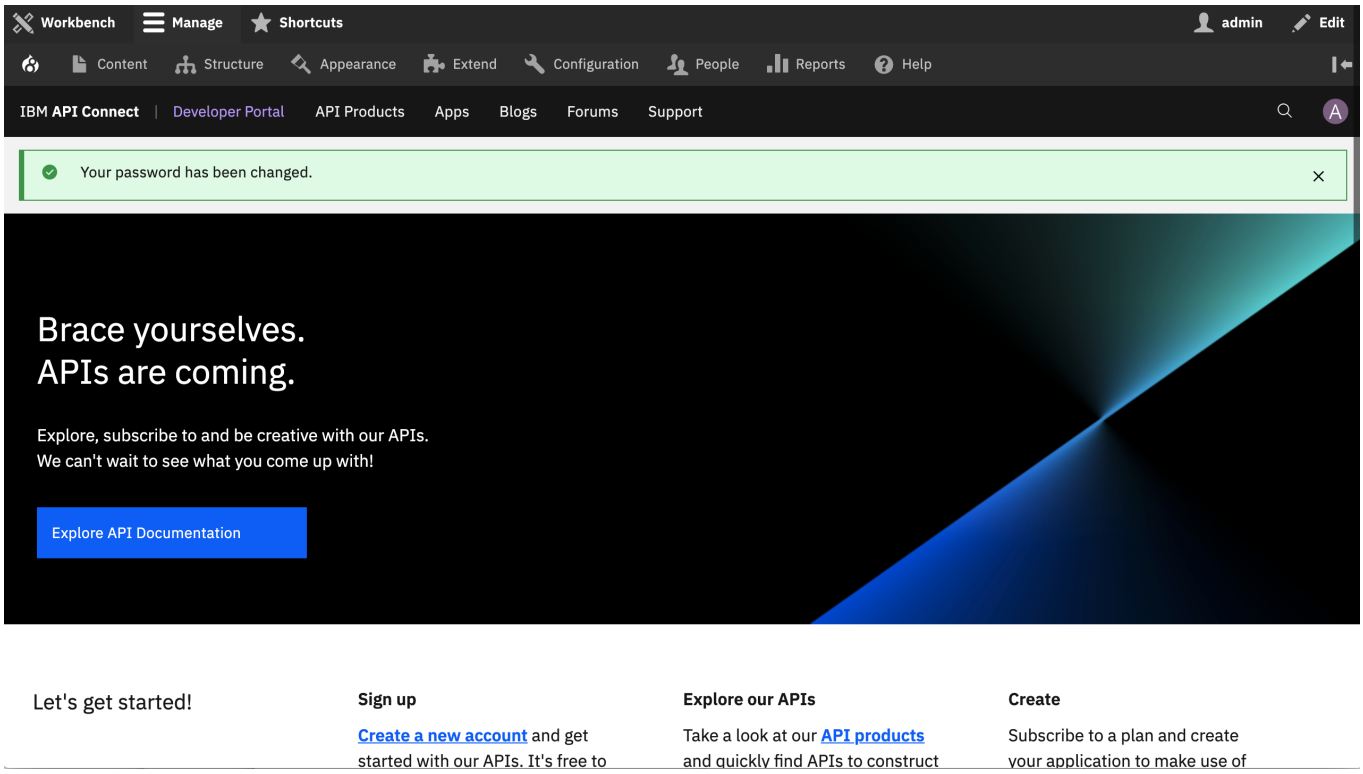
11. Click the email link. Click Sign in.



12. Change the password. Click Submit.



The portal site for the Production catalog is now active. The admin account cannot create applications or subscriptions. You must create a new account to complete those tasks.



What you did in this tutorial

In this tutorial, you completed the following activities:

- Enabled a Developer Portal site and created an administrator account.

Note: To ensure that your desired content is displayed, it is crucial to publish your product containing APIs to the appropriate catalog.

What to do next

- Create a Developer Account and Application

Related information

- Importing an API

Tutorial: Creating a private Developer Portal site

You can create a private version of the Developer Portal, so that authenticated users only can access content on the site. You might want to do this if you want to restrict public access to your content unless they are logged in.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

In this tutorial, you are going to restrict access for all the content to authenticated users only. This gives access to all users from **`Content Author`** to **`Superuser`**, as they are all required to authenticate to access their roles. You will then add the login form to the home page, so that public users will be able to view the public home page that contains the login form only.

This tutorial takes you through the following steps:

1. Export your configuration settings.
2. Restrict all permissions to authenticated users.
3. Restrict blocks to authenticated users.
4. Add login form to the home page.

Export your configuration settings

In case you make an error, or want to revert all of the changes made during this tutorial, you can back up your configuration settings to your machine before you make any changes.

The tutorial Exporting and importing custom themes and site configuration provides further details on how to export your configuration settings using the Developer Portal CLI.

Restrict all permissions to authenticated users

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to People > Permissions.
3. Clear all check boxes that are associated with **ANONYMOUS USER**.
4. Click **Save permissions** at the end of the page.

Restrict blocks to authenticated users

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Block layout.
3. Find the row that is labeled **Main menu** and click **Configure**.
4. Find the setting **Roles** under the category **Visibility**.
5. Check the box **Authenticated user**.
6. Click **Save block**.
7. Repeat from step 3 for the **Footer menu**.
8. Click **Save block**.

Add login form to the home page

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages.
3. Click **Edit** on the **Welcome** page.
4. Navigate to Panels > Content.
5. Click **Add new block**.
6. Find and click **User login**, under the **FORMS** section.
7. Clear the **Display title** check box.
8. Click **Add block**.
9. Rearrange the new login block so that it is underneath the **Welcome Banner**.
10. Click **Update and save**.

What you did in this tutorial

On your Developer Portal, you restricted access for all the content to authenticated users only. Now when you access your home page as an unauthenticated user, you see that the login form appears underneath the welcome banner. Navigating to other URLs on the site presents an **unauthorized** page to appear.

When you log in, the login form disappears from the home page and you are able to access all of the pages that are visible to your role.

Tutorial: Creating Accounts and Applications on the Developer Portal

This tutorial shows you how to create a developer account and register an application on the Developer Portal.

Before you begin

You must have a Developer Portal installed, and created and published at least one API product.

If you do not have a Developer Portal available, complete the tutorial Creating a Developer Portal.

If you don't have an API product available, you can download one of the following APIs, and then import and activate it:

- If your Sandbox catalog uses a DataPower® Gateway (v5 compatible) (as one of the gateways, or as the only gateway), download the findbranch.txt file to your local filesystem. Rename this file `findbranch.yaml`.
- If your Sandbox catalog uses a DataPower API Gateway, download the findbranch_v6.txt file to your local filesystem. Rename this file `findbranch.yaml`.

For more information about importing an API, see Adding a REST API by importing an OpenAPI definition file.

About this task

In this tutorial you are going to complete the following lessons:

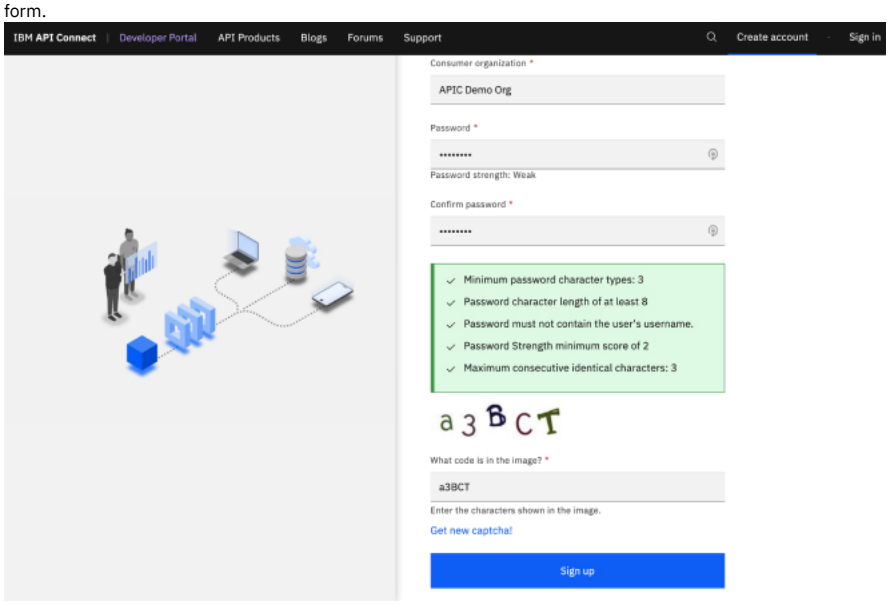
- Creating a developer account in the Developer Portal
- Creating a developer application in the Developer Portal
- Testing the Branches API in the Developer Portal

Note: The administrator account cannot create an App, or register to an App. To create or register to an App, you must have a developer account. A developer account that has been assigned administrator privileges can create and register Apps.

Creating a developer account in the Developer Portal

Take the following steps to create a developer account:

1. Follow the instructions provided on Creating a Developer Portal and open the Developer Portal.
2. Click Create account. If the Create account link is not visible, log out as Admin from the Developer Portal first, go to the home page of the Developer Portal, and click Create account.
   Note: Your email is used as your user name for the Developer Portal. The email address must be different from the address used for creating the administrator account.
3. Complete the sign up form. The email address you provide must be functional. Click Sign up. A confirmation message is sent to the email address provided in the form.

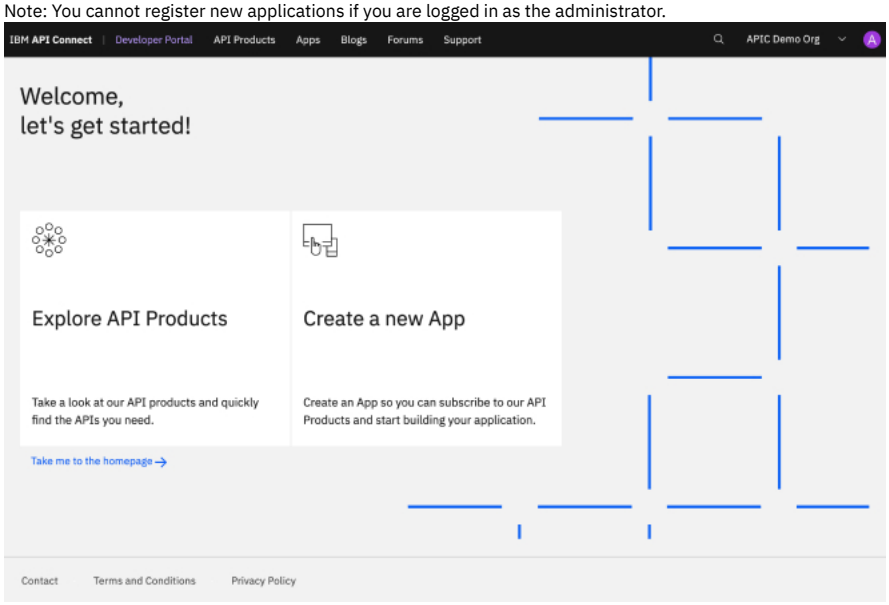


4. Click the account activation link in the confirmation email to activate your account.
5. To use the Developer Portal, click Sign in and sign in with the user credentials you specified.

Creating a Developer Portal Application

To create and register a new App:

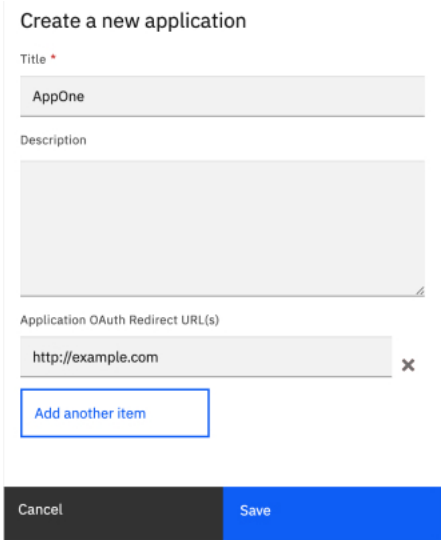
1. In the Developer Portal, click Create a new App.
   Note: You cannot register new applications if you are logged in as the administrator.



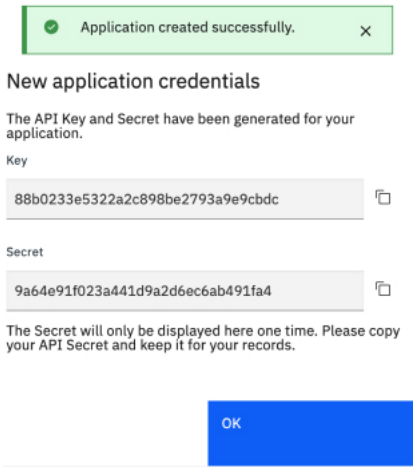
2. Enter the following values for the application that is being registered.

Table 1. Values for registering the application

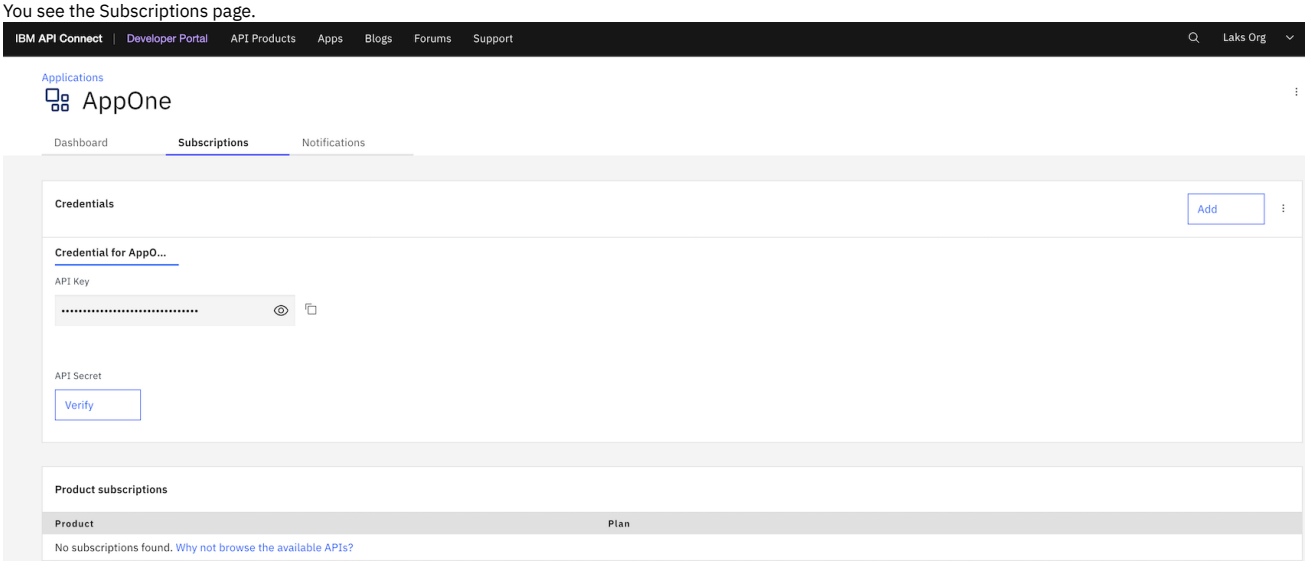
Field Name	Value
Title	`AppOne`
Application OAuth Redirect URL(s)	`http://example.com`



3. Click Save.
4. Copy the key and secret values for the new app. Click OK.



5. Click OK.
   You see the Subscriptions page.



Subscribing to an API in the Developer Portal

1. Click API Products.
2. In the Name field, enter `FindBranch` and click on the *product name* .
3. Under Plans, click Select.
4. Under the Application heading, click AppOne.
5. Under Subscribe, click Next to confirm the subscription.

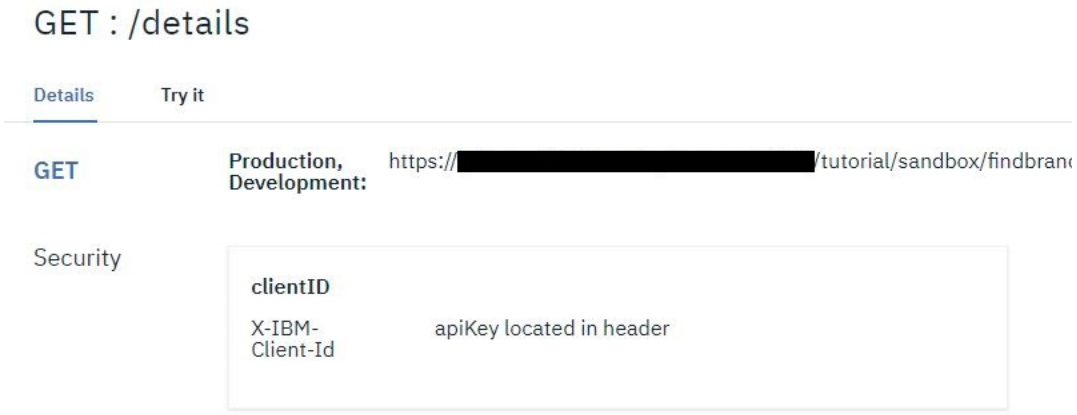
6. Under Summary, click Done after confirming the subscription details.

To view the current subscription details of your application, perform the following steps:

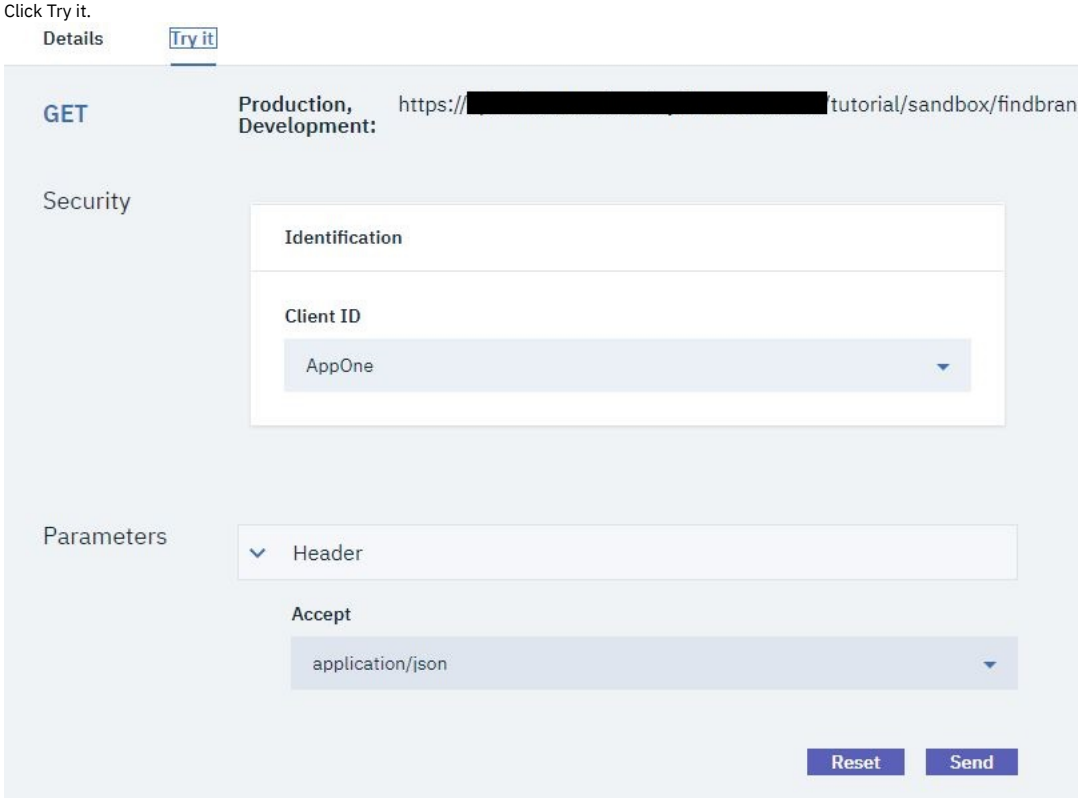
1. Under Apps, select AppOne.
2. Click Subscription.

Testing An API in the Developer Portal

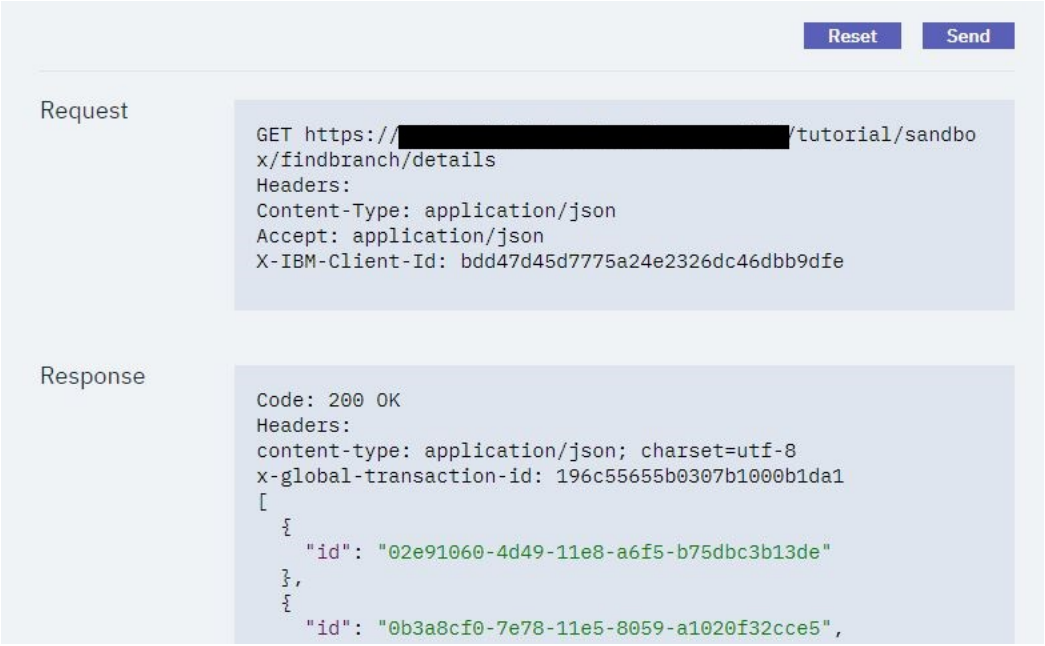
1. Click API Products in the Developer Portal dashboard.
2. Click a Product, such as FindBranch auto product , then select the FindBranch API from the list.
3. Click GET /details.



4. Click Try it.



5. Click Send.
   Note: If no response is received, navigate to the URL that is displayed at the beginning of the Try this operation section, in a new browser tab. Accept the security certificate, and then call the operation again.
6. A returned response of `200 OK` and the message body are displayed, indicating that the REST API operation call was successful.



What you did in this tutorial

In this tutorial, you completed the following activities:

- Created a developer account in the Developer Portal.
- Created and registered a new App, and subscribed it to a Plan.
- Tested an API in the Developer Portal.

Tutorial: Creating a custom theme for the Developer Portal

Customize the appearance of your Developer Portal home page, by generating a sub-theme, configuring the overrides.css file, and installing the newly customized theme.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.
Important: You're not permitted to include any IBM® API Connect code within any custom themes that you create.

About this tutorial

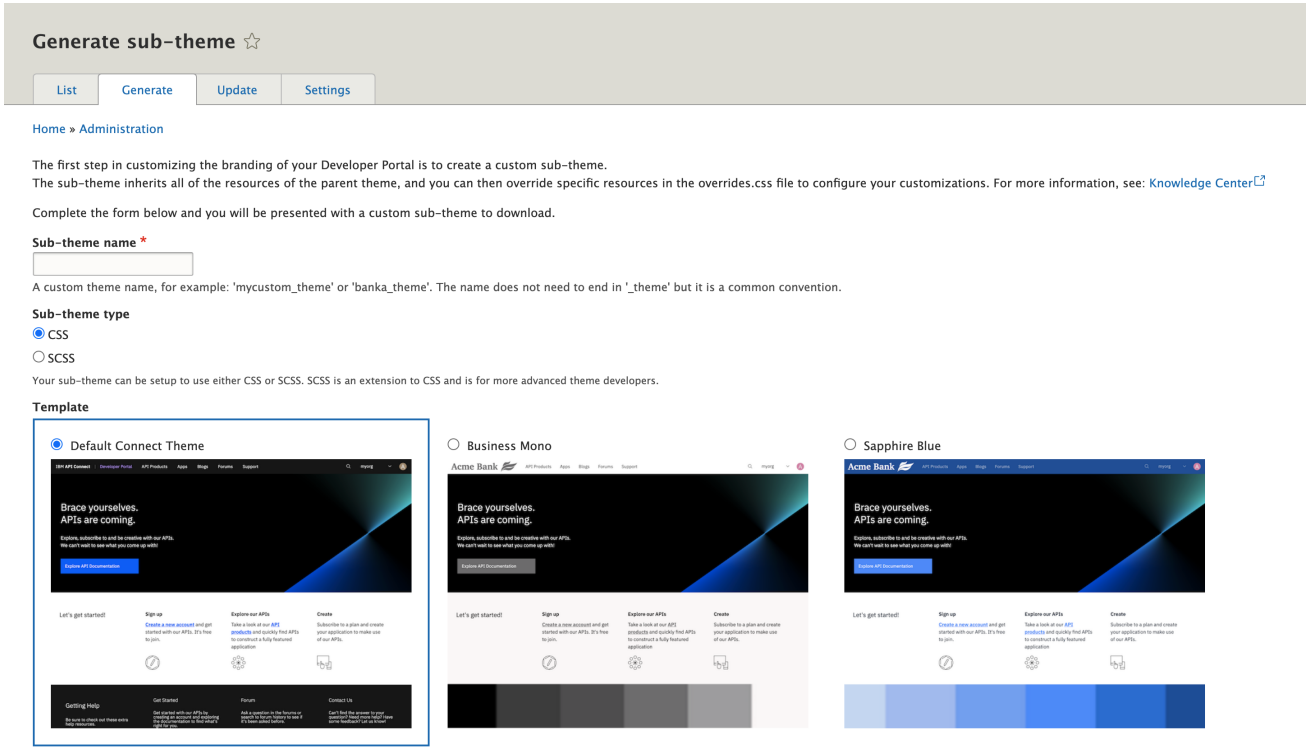
The way to create a custom theme is to create a custom sub-theme of the standard API Connect theme that the Developer Portal uses by default. The sub-theme inherits all of the resources of the parent theme, and you can then override specific resources to configure the required customizations.

This tutorial takes you through the following lessons:

1. Creating a sub-theme by using the theme generator.
2. Customizing the overrides.css file.
3. Installing and enabling your customized theme.

Creating a sub-theme

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Appearance > Generate sub-theme. The Generate sub-theme window is displayed.
4. Enter a Sub-theme name, and select CSS for the Sub-theme type. (If you prefer, you can select SCSS, but this extension to CSS is for advanced theme developers, and isn't covered by this tutorial.)



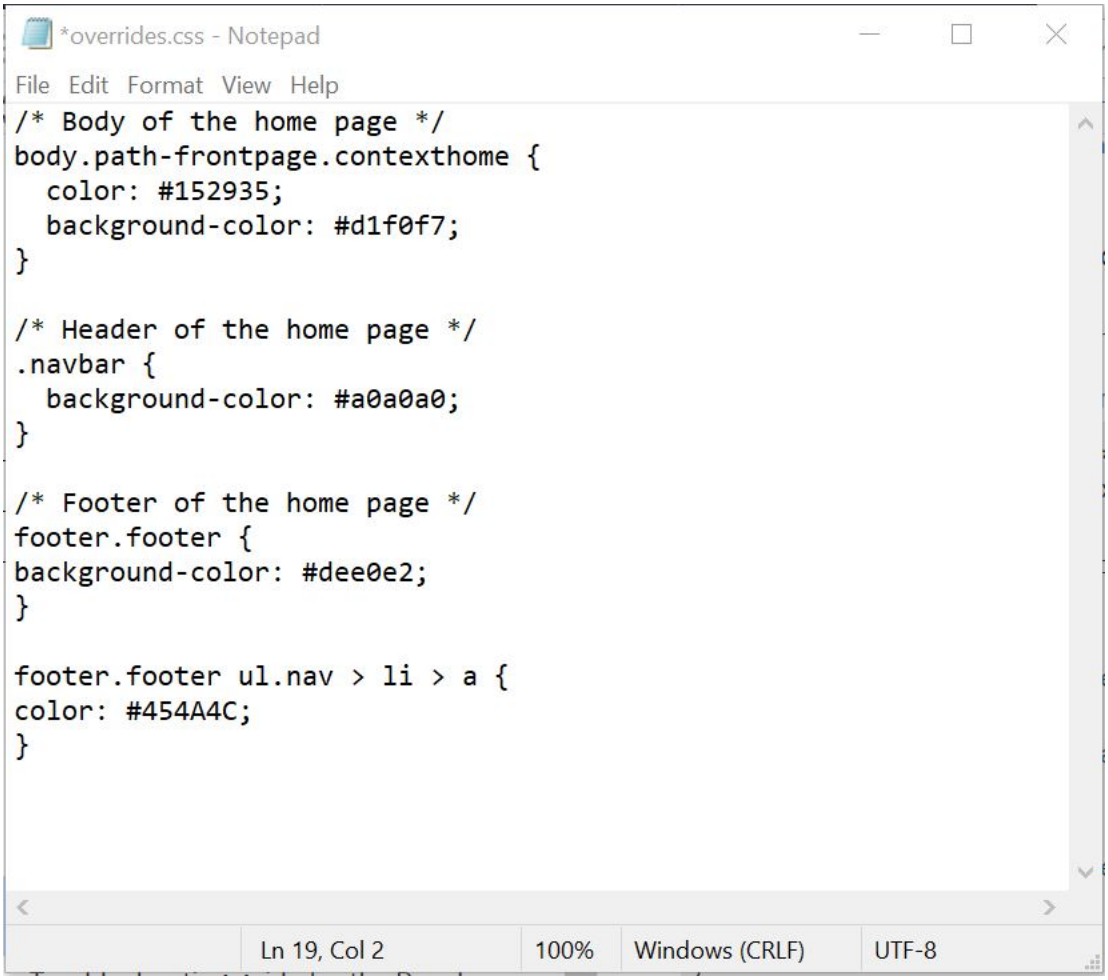
5. Select the Default template to base your **`sub-theme`** on. You can create a sub-theme based on a color template. However, for this tutorial you use the Default Connect Theme template.
6. Click Generate.
7. Download the generated sub-theme to a location of your choice, and extract all the files from the .zip file.

Customizing the overrides.css file

On your computer, navigate to the files that you extracted. Find the overrides.css file in here: <extract_folder>/css/overrides.css. You can now customize the overrides.css file.

1. Open the overrides.css file in your chosen editor.
2. Customize your sub-theme by entering the following elements into the overrides.css file:

```
/* Body of the home page */
body.path-frontpage.contexthome {
  color: #152935;
  background-color: #d1f0f7;
}

/* Header of the home page */
.navbar {
  background-color: #a0a0a0;
}

/* Footer of the home page */
footer.footer {
background-color: #dee0e2;
}

footer.footer ul.nav > li > a {
color: #454A4C;
}
```

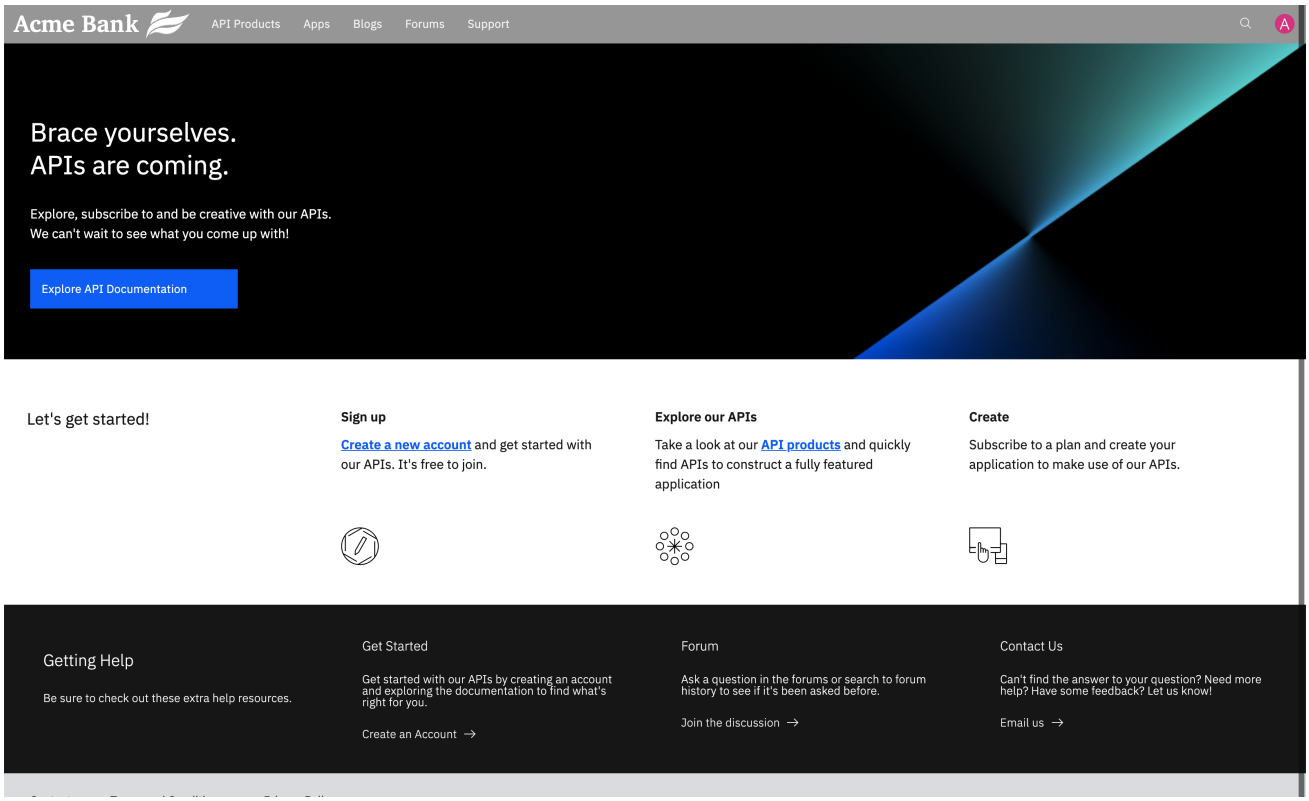


3. Save the overrides.css file.

Installing and enabling your customized theme

1. After you have finished updating the overrides.css file, compress all the theme files back into the .zip sub-theme file that you downloaded originally.
   Note that you can also include your own custom logo and favicon in your theme files, and these are then included as part of your theme when it is installed. These files must use the naming convention logo.svg, and favicon.ico, respectively. You can also change your logo and favicon at a later time, see Changing the site logo and Changing the shortcut icon.

2. In the Developer Portal, click Appearance > Install new theme. The **`Install new theme`** window is displayed.
3. In Upload a module or theme archive, click Browse, and navigate to your newly updated compressed theme file.
4. Click Continue to install the theme onto your site.
5. Click Enable newly added themes, and find your new theme in the list of Disabled themes. Click Enable and set as default to set your new custom sub-theme as the default theme for your site.
6. Return to the Developer Portal home page by clicking Back to site. You can now see your custom theme.



What you did in this tutorial

In this tutorial, you completed the following tasks:

- Created a sub-theme of your Developer Portal site by using the theme generator.
- Configured some customized home page elements in the overrides.css file of your sub-theme.
- Uploaded and installed your customized theme in your Developer Portal site.
- Successfully customized the appearance of your Developer Portal home page.

Tutorial: Using avatars

You can use an avatar generator to better represent the branding of your site, and add a sense of community for your users, than the default name initial that is provided in the Developer Portal.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

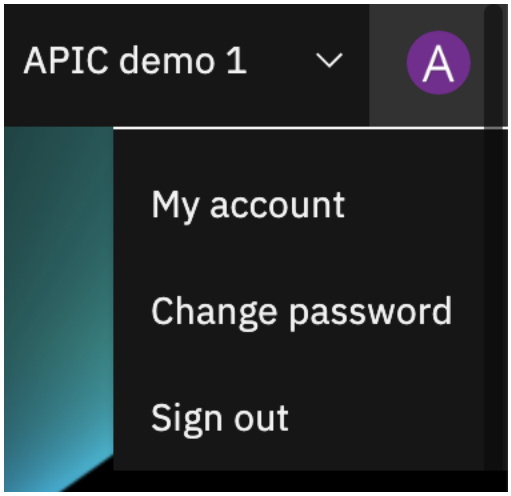
About this tutorial

In this task, you add the **Robohash** avatar generator to your Developer Portal site, and assign it as the primary avatar generator for all users.

This tutorial takes you through the following steps:

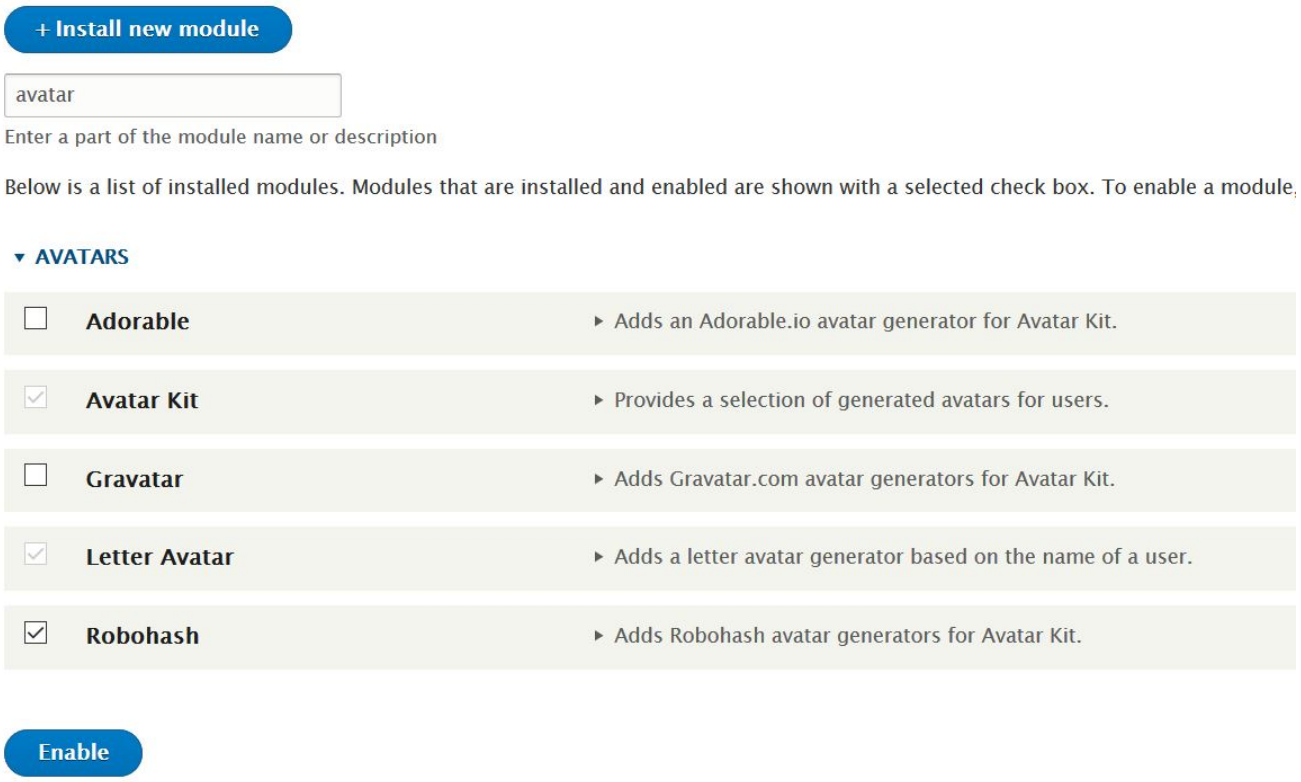
1. Install the **Robohash** avatar generator.
2. Set **Robohash** as the preferred avatar generator for your site.
3. Configure the permissions for the **Robohash** avatar generator.

The avatar of the admin user displays like this before the block is added.



Enable the `Robohash` avatar generator

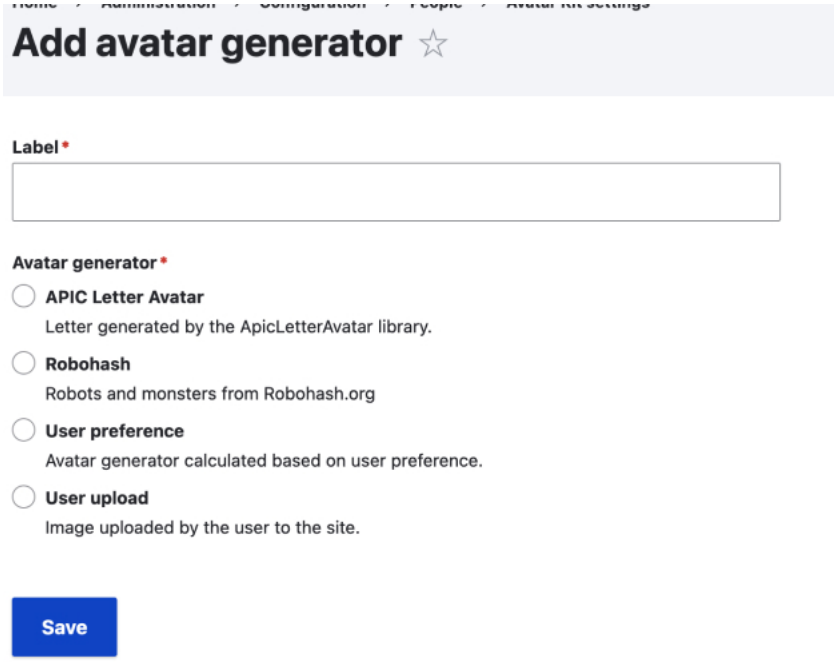
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend.
4. In the text box, enter `avatar` in the **`Block description`**.
5. Check the **`Robohash`** box, click Enable.



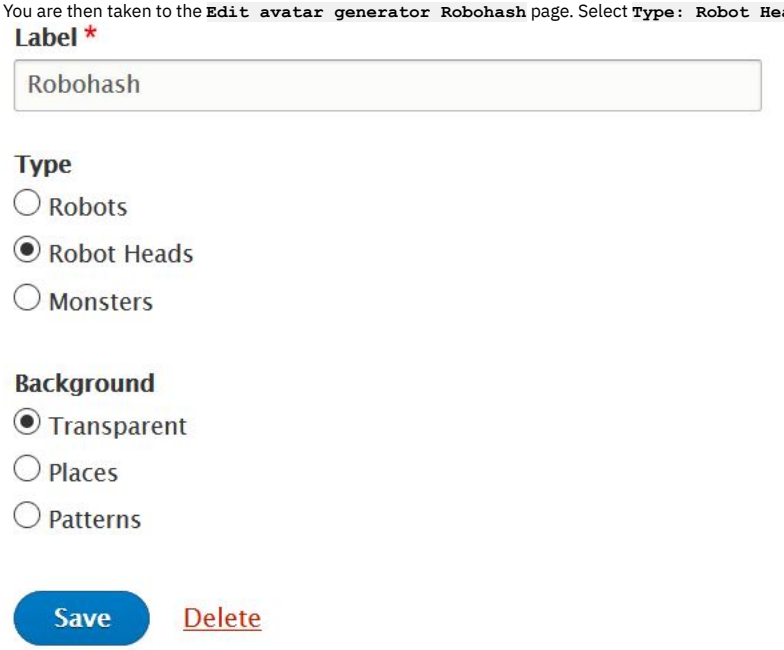
When the module is enabled, you receive a **`Module Robohash has been enabled`** message.

Set `Robohash` as the preferred avatar generator for your site

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > People > Avatar settings.
3. Click **`+ Add avatar generator`**.
4. Enter `Robohash` as the label.
5. Check **`Robohash`** for the avatar generator.
6. Click Save



7. You are then taken to the **`Edit avatar generator Robohash`** page. Select **`Type: Robot Heads`** and **`Background: Transparent`**. Click Save.



8. You are now on the avatar generator page. Drag **`Robohash`** to be first in the list, and check **`Enabled`**, then click Save configuration.

You have now set **`Robohash`** as the preferred avatar generator for your site.

Configure the permissions for the `Robohash` avatar generator

The default setting for the **`Robohash`** avatar generator is to generate avatars for users with **`superuser`** access only. In this example, the permissions are changed so that all **`authenticated users`** can access **`Robohash`**.

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to People > Permissions.
3. Find **`Use Robohash`** under the permission type **`Avatar Kit`**.
4. Check **`AUTHENTICATED USER`** for **`Use Robohash`** to enable the generator for all authenticated users.
   Notice that all the boxes for this user type are then automatically checked because all of those users have the same permissions as an authenticated user, or higher.

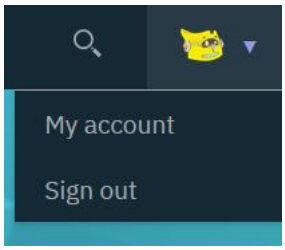
PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	FORUM MODERATOR	CONTENT AUTHOR	ADMINISTRATOR	SUPERUSER
Change own logout threshold	☐	☐	☐	☐	☑	☑
Avatar Kit						
Administer Avatar Kit *Warning: Give to trusted roles only; this permission has security implications.* Manage settings for Avatar Kit.	☐	☐	☐	☐	☑	☑
Use *Letter Generator* User can select *Letter Generator* avatar generator.	☐	☑	☑	☑	☑	☑
Use *Robohash* User can select *Robohash* avatar generator.	☐	☑	☑	☑	☑	☑

5. Click Save permissions.

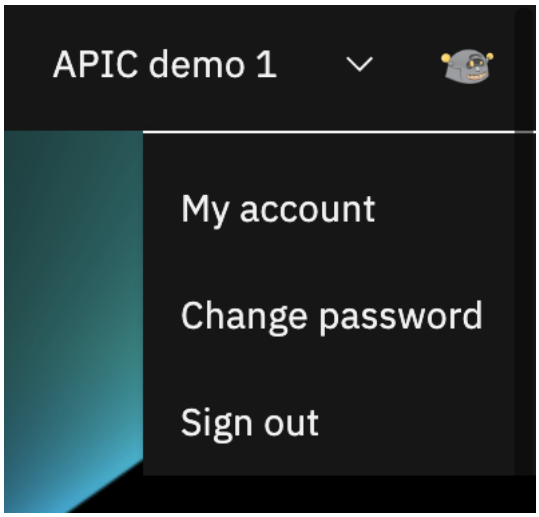
What you did in this tutorial

The avatars of all users now have the **robohash** avatar.

You can check that the avatar of the admin user has changed by looking at your site page.



You can check that the avatars of other users have changed by finding their published content and looking for their avatar. Or, you can navigate to Manage > People, click a user, and see their avatar on their profile page.



What to do next

You can disable the **robohash** avatar generator.

1. Navigating to Extend > Disable module.
2. Search for **avatar** in the filter text box.
3. Check **Robohash**, click Disable
4. Click Disable.

You can also disable the **robohash** avatar generator by clearing it within Configuration > People > Avatar settings, or, reducing its preference by dragging it to a different order in the list. Click Save configuration.

Tutorial: Using image optimization

You can use image optimization on your Developer Portal site to enhance the use of images.

Optimized images do not overload your website and significantly increase its performance compared to using the original images. Your customer satisfaction and brand reputation improve with fast-loading and trimmed visual assets. Automatic optimization can save time and effort on editing and content moderation.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

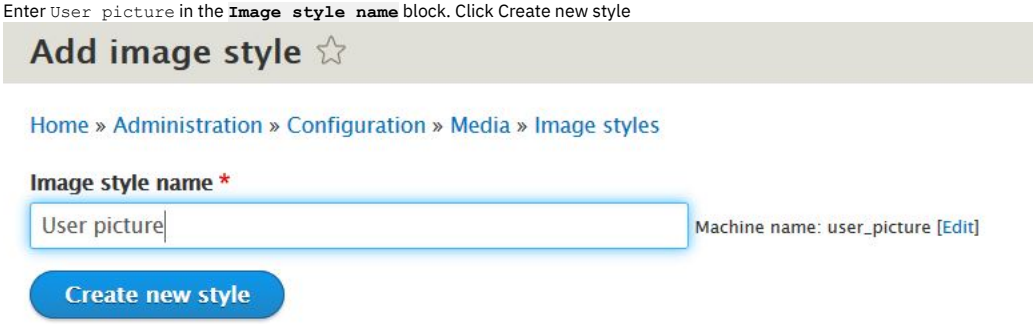
In this task, you create an image style that is called `User picture`, and give it a scale and crop effect. Then, use the `User picture` as the **`Picture`** field in the user account.

This tutorial takes you through the following steps:

1. Create the image style **`User picture`**.
2. Edit the image style **`User picture`**.
3. Assign **`User picture`** to the **`picture`** option in **`Account settings`**..

Create the image style `User picture`

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > Media > Image styles.
4. Click + Add image style.
5. Enter `User picture` in the **`Image style name`** block. Click Create new style

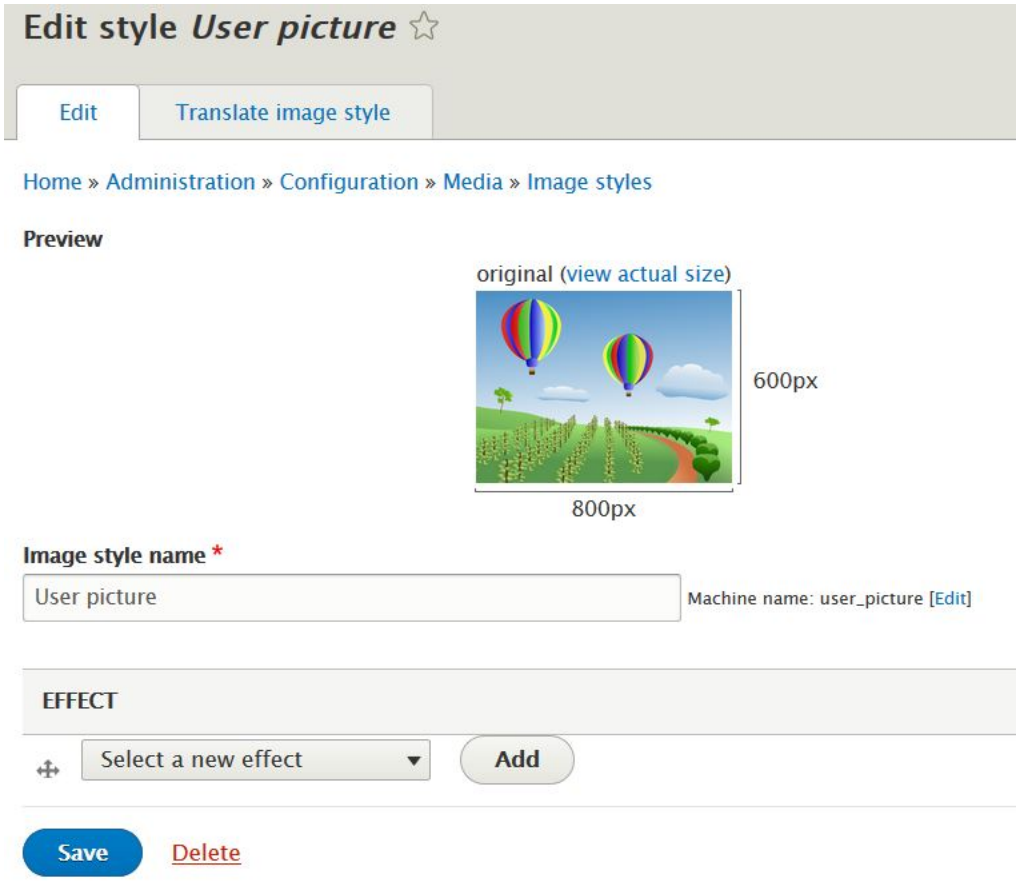


You created the image style **`User picture`**.

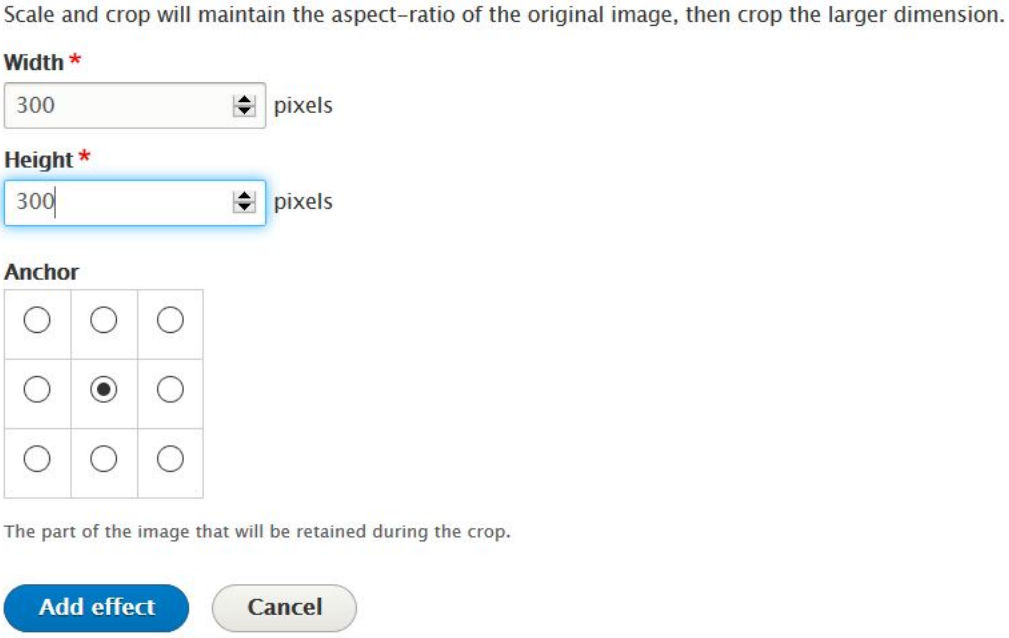
Edit the image style `User picture`

When you edit this style, you choose the width and height settings. Also, the part of the picture to retain during the crop, for example, the center.

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > Media > Image styles.
3. Click Edit on the **`User picture`** style name.



4. Select **`Scale and crop`** from **`Select a new effect`**, then click Add.
5. Enter `300` in the **`Width`** and **`Height`** boxes. Leave the **`Anchor`** as the middle setting. Click Add effect.



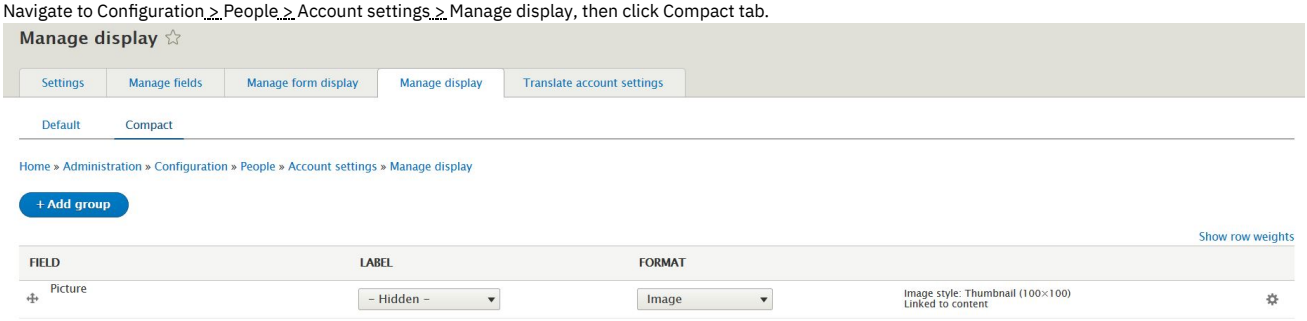
6. Click Save.

You edited the image style **`User picture`**.

Assign `User picture` to the `picture` option in `Account settings`

1. If the administrator dashboard isn't displayed, click Manage to display it.

2. Navigate to Configuration > People > Account settings > Manage display, then click Compact tab.



3. On the **`Picture`** row, click ⚙.
4. Select **`User picture`** from the **`Image style`** drop-down.
5. Click Update and Save.

You set the **`User picture`** image style for all accounts.

What you did in this tutorial

You changed your account settings so that if the user displays a picture it is within the limits of the image style **`User picture`**.

To check the picture used by a user, navigate to Manage > People, click a user, and see their picture on their profile page. Right-click the image, then click **`View image info`**, you can see that the dimensions are scaled.

Type:	JPEG Image
Size:	29.95 KB (30,667 bytes)
Dimensions:	620px × 413px (scaled to 240px × 160px)
Associated Text:	Profile picture for user apicdemo1

Tutorial: Adding a Back to top button

You might want to add a Back to top button, as a navigation enhancement to help your Developer Portal users get back to the top of the page that they are viewing, instead of scrolling.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

About this tutorial

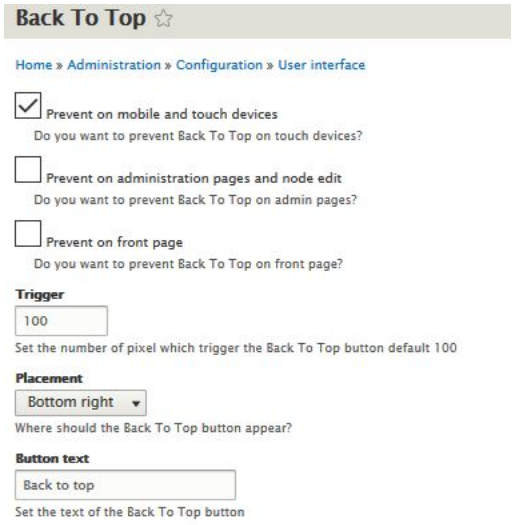
In this tutorial, you add a Back to top button to your Developer Portal.

Before you begin, ensure that the **`Back to top`** module is installed and enabled:

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend.
4. In the text box, enter `back to top` in the **`Block description`**.
5. Check the **`Back To Top`** box, click Enable.

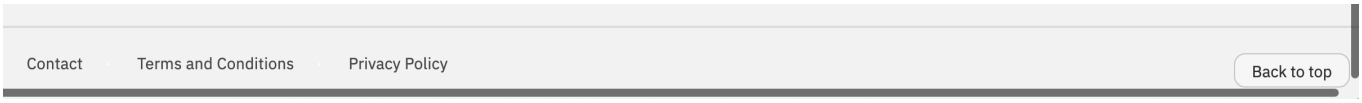
Configure the `Back to top` module

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > User interface > Back to Top.



3. Change any settings as required, or keep the defaults.
4. Click Save configuration

You now have a Back to Top button.



What you did in this tutorial

You added a Back to Top button, as a navigation enhancement to help your users get back to the beginning of a page that they are viewing, instead of scrolling.

Tutorial: Creating a drop-down menu link

You can create and use drop-down menus, or nested menus, to enhance the use of your Developer Portal .

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

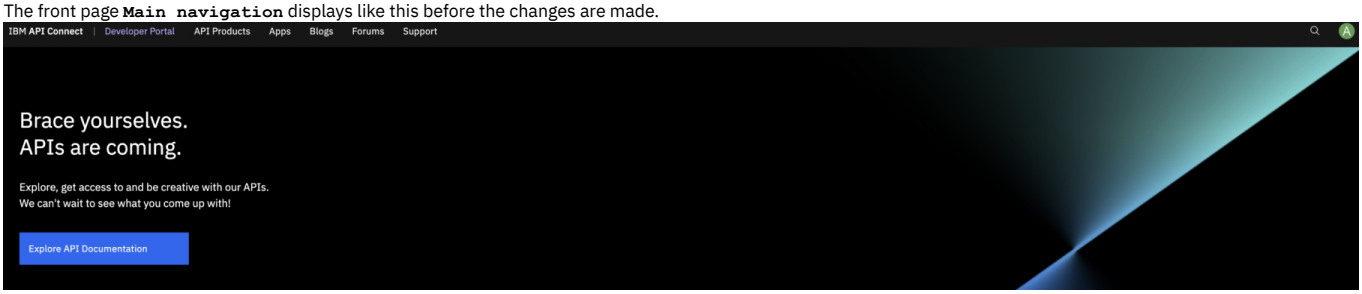
In this tutorial, you create a drop-down menu link on the **`Main navigation`** menu.

Note:
Although drop-down menu links can be created in the Developer portal, the drop-down menu doesn't expand by default. To make the drop-down menu expand, you need to use the **`Superfish`** module.

This tutorial takes you through the following steps:

1. Add menu links to the main navigation menu.
2. Make the drop-down menu expand by using the Superfish module.

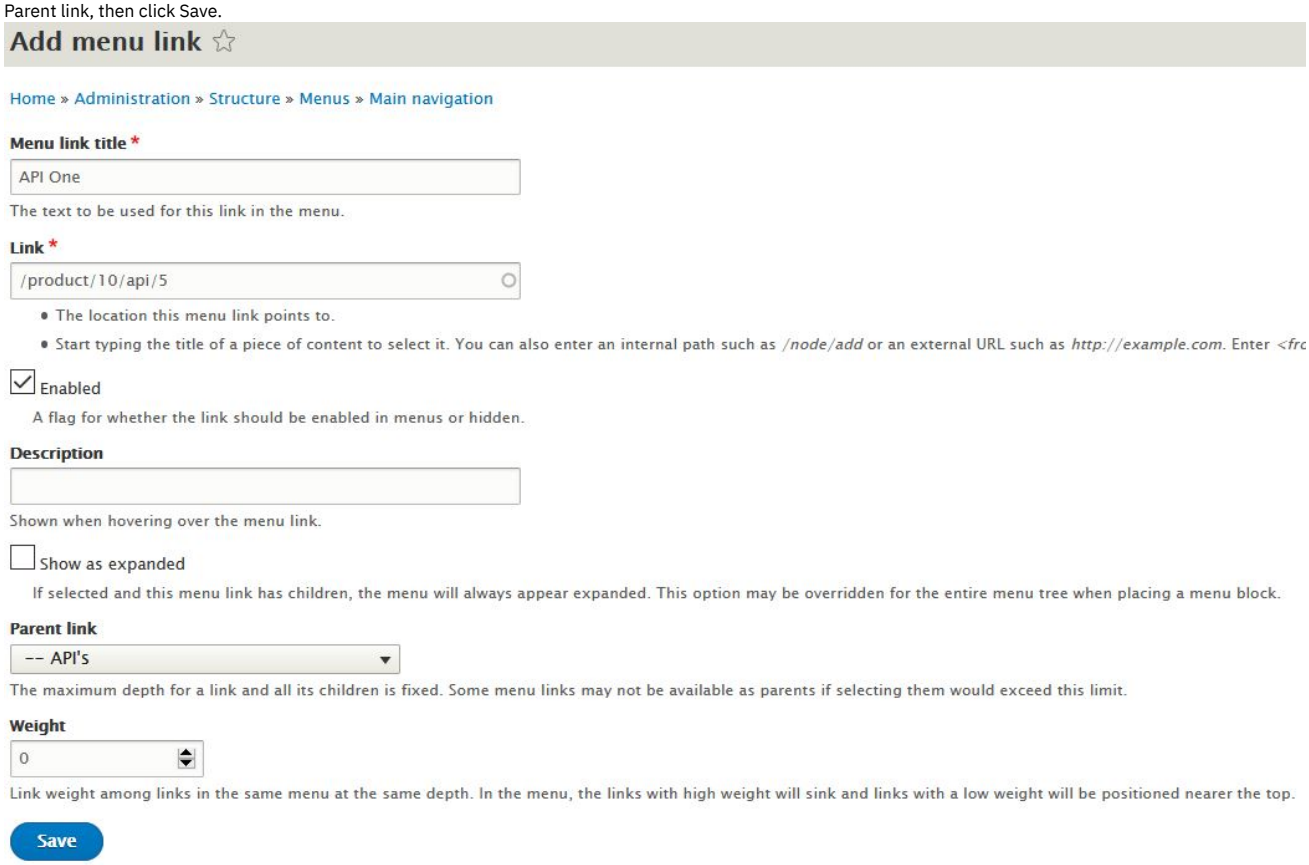
The front page **`Main navigation`** displays like this before the changes are made.



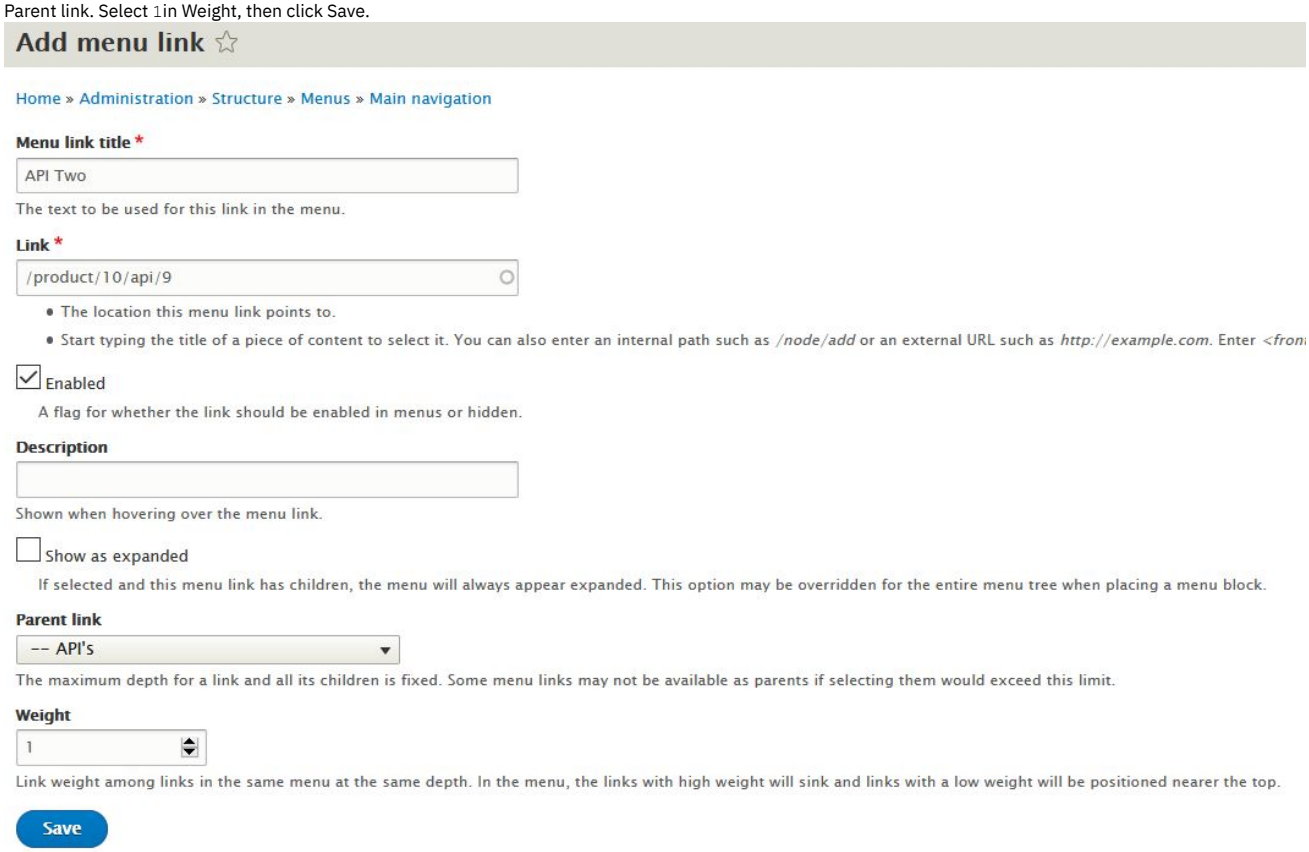
Add menu links to the main navigation menu

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.

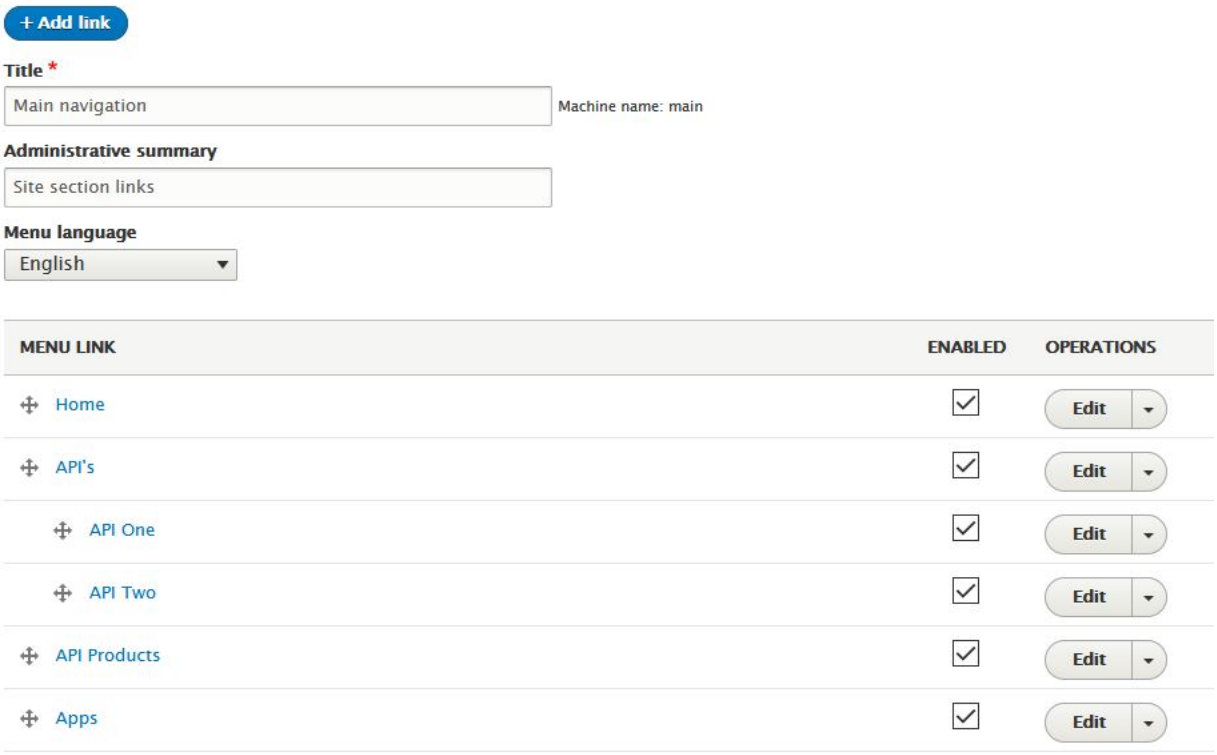
3. Navigate to Structure > Menus > Main navigation > Add link
4. Enter `API's` in the **`Menu link title`**. Enter `/api` in the **`Link`**. Check Show as expanded, then click Save.
   Note: Ensure Show as expanded is checked as this is the parent link for the submenu links.
5. Add another menu link to the main navigation menu. Enter `API One` in the **`Menu link title`**. Enter `/product/10/api/5` in the **`Link`**. Select -- API's in the Parent link, then click Save.



6. Add another menu link to the main navigation menu. Enter `API Two` in the **`Menu link title`**. Enter `/product/10/api/9` in the **`Link`**. Select -- API's in the Parent link. Select `1`in Weight, then click Save.



7. Navigate to Structure > Menus > Main navigation. The hierarchy of the menu links now looks like this:

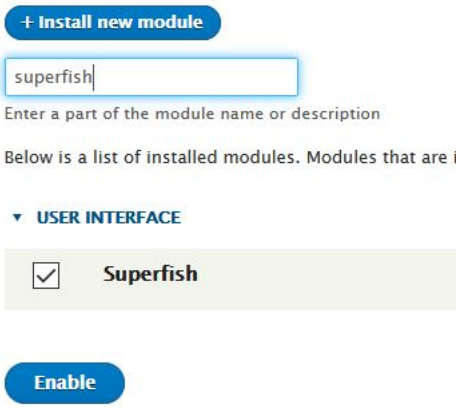


8. Navigate to the home page. The **`APIs`** menu link that you added to the main navigation menu does appear, but the submenus links don't expand when you hover over it.

Make the drop-down menu expand by using the Superfish module

To make the drop-down menu expand when you hover over it, you need to use the **`Superfish`** module and configure the **`Block Layout`**.

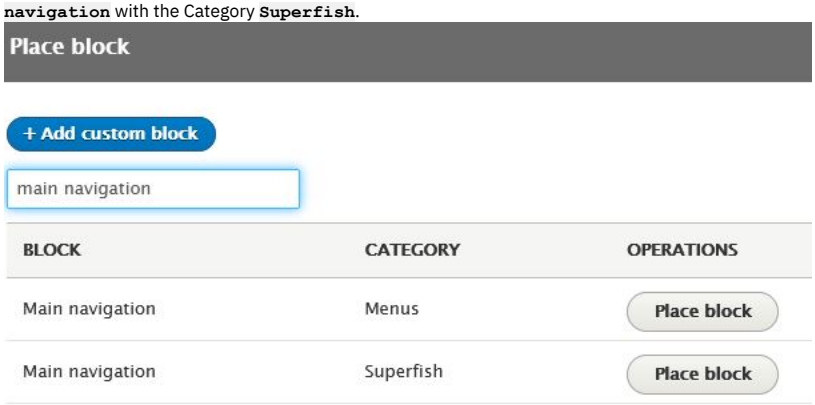
1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Extend. Search for **`superfish`**, check the **`Superfish`** box, click Enable.



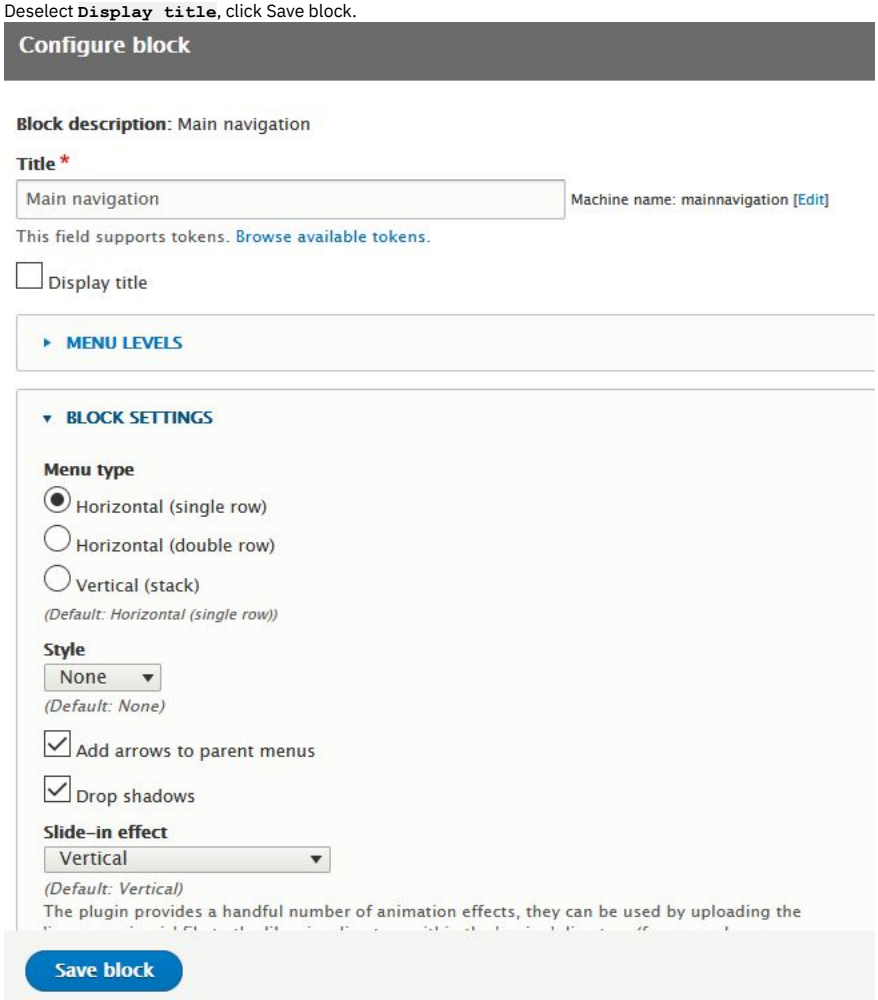
3. Navigate to Structure > Block layout.

4. Click Configure, and select Disable for the **`Main menu`** block.

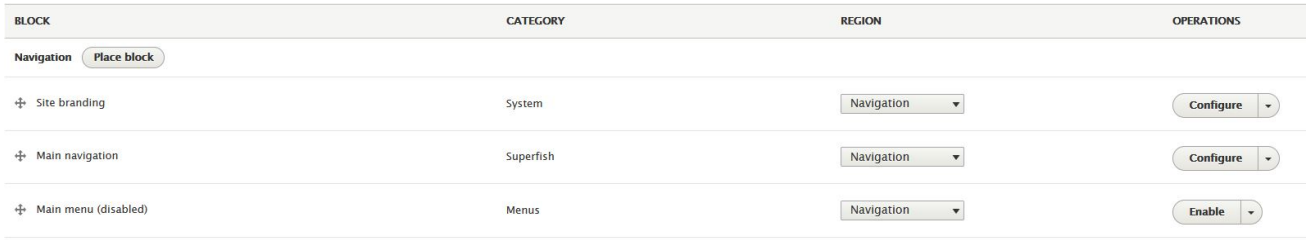
5. Click Place block for Navigation, Then search for `main navigation` in the box. The filtered results might show two or more options. Click Place block for the **`Main navigation`** with the Category **`Superfish`**.



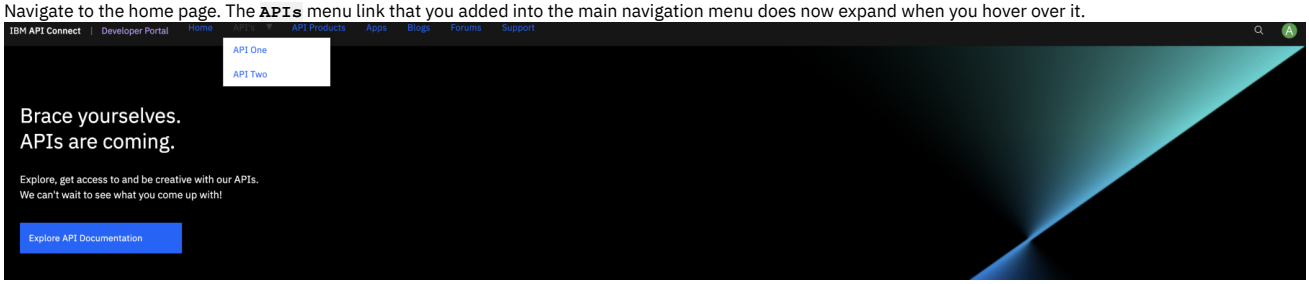
6. Deselect **`Display title`**, click Save block.



7. Drag **`Main navigation`** to be under **`Site branding`**.



8. Click Save blocks.
9. Navigate to the home page. The **`APIs`** menu link that you added into the main navigation menu does now expand when you hover over it.



What you did in this tutorial

You created a drop-down menu link on the **`Main navigation`** menu.

What to do next

You can consider any other uses of the drop-down menu to enhance your users experience on your Developer Portal.

Tutorial: Displaying blog posts on the front page

You can customize the appearance of the front page of your Developer Portal to display recent blog posts. You might use the blog posts to provide your users with information, such as service availability or marketing campaigns.

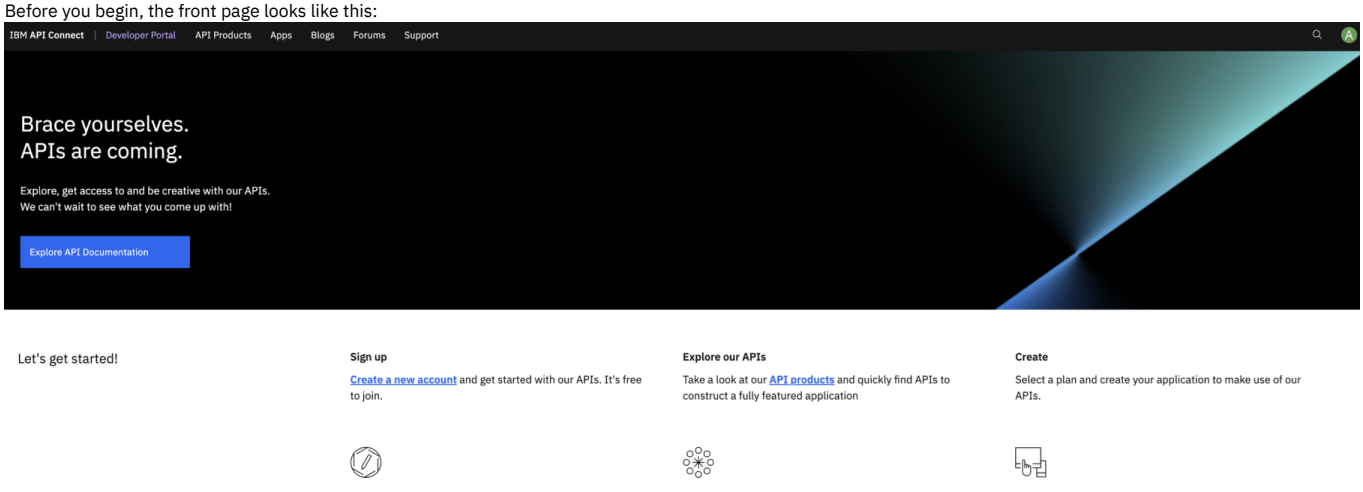
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

In this task, you create a view, customize that view to show blog posts, then embed that view on the front page of your Developer Portal site.

Before you begin, the front page looks like this:

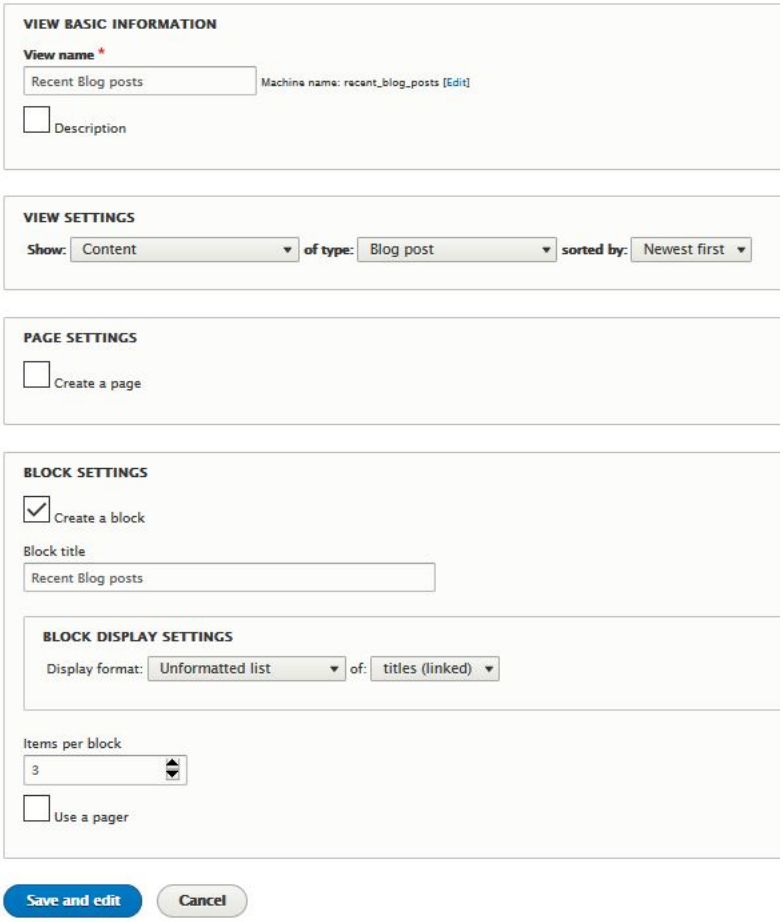


This tutorial takes you through the following steps:

1. Create a view.
2. Customize the view.
3. Embed the view on the front page of your Developer Portal.

Create a view

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Views > Add view.
4. In the **View name** block, enter a name for your view, for example Recent Blog posts.
5. Under **VIEW SETTINGS**, show **Content** of type **Blog Post** sorted by **Newest first**.
6. Check **Create a block** in **BLOCK SETTINGS**, and choose the number of blog posts that you want to be displayed.



7. Click Save and edit.

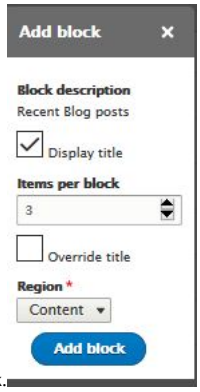
Customize the view

You are now on the page where you can edit your view.

1. If you are not on the edit view page, navigate to Structure > Views, find the name of the view that you created, and click Edit.
2. Click Unformatted list in the **Format** section.
3. Select **Bootstrap List Group**, then click Apply.
4. Select Content: Title in the **Title field** drop-down, then click Apply.
5. Click Save.

Embed the view on the front page of your Developer Portal

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages.
3. Click **Edit** on the welcome page.
4. Click **Panels**, then **Content**.
5. Click +Add new block.
6. Select your new block, which is under **LISTS (VIEWS)**.



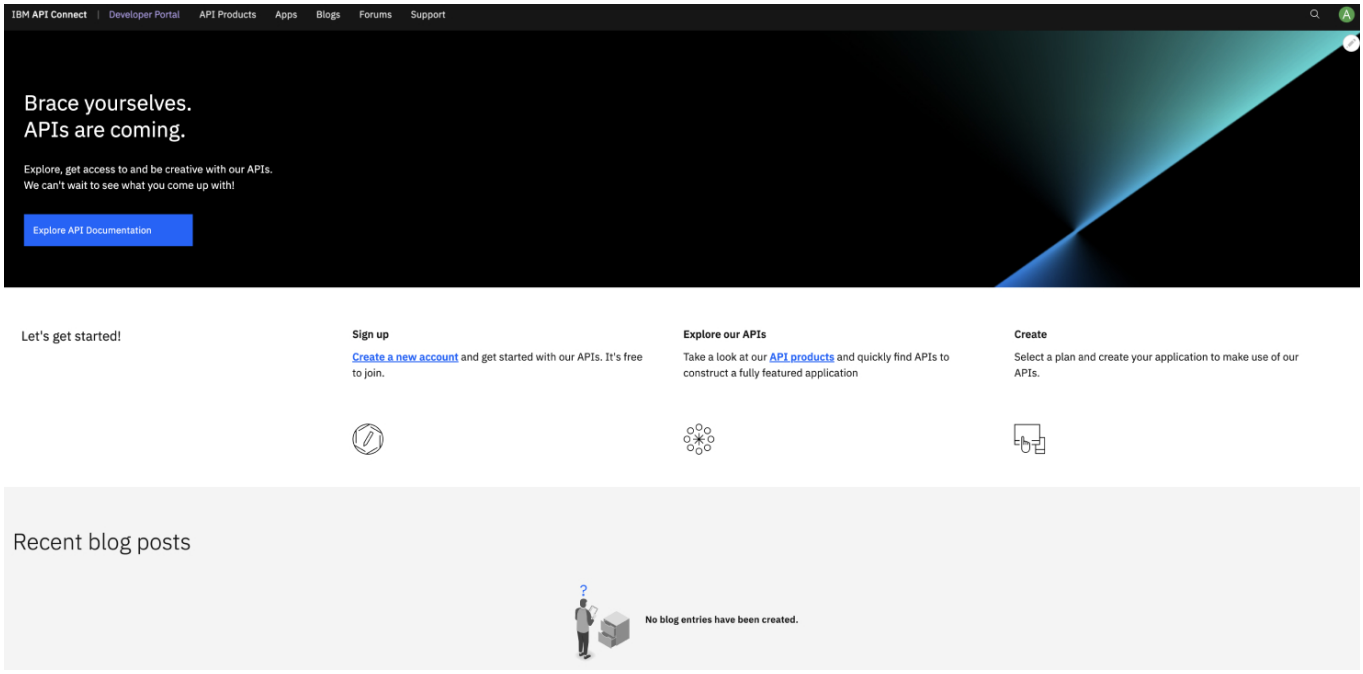
7. Select the number of **`Items per block`**, then click Add block.
8. You can now rearrange your view to be where you would like it on the front page.

LABEL	PLUGIN ID
Content	
Welcome Banner	block_content:2fcdb632-d4b9-44d9-b195-359e751966a1
Featured Content	featuredcontent
Recent Blog posts	views_block:recent_blog_posts-block_1
Getting Started	block_content:f2b7762b-97f4-4d51-b7dc-d6de5be6d0ce

9. Click Update and save.

What you did in this tutorial

You can check that the view you created appears on the front page of your site page.



What to do next

You can edit your view at any time by navigating to Structure > Views and selecting your view by its name. You can also modify or delete the view's layout on the front page by navigating back to the layout section of your page under Structure > Pages.

Tutorial: Displaying tweets on the front page

You can customize the appearance of the front page of your Developer Portal to display recent tweets. You might use the tweets to provide your users with information, such as service availability or marketing campaigns.

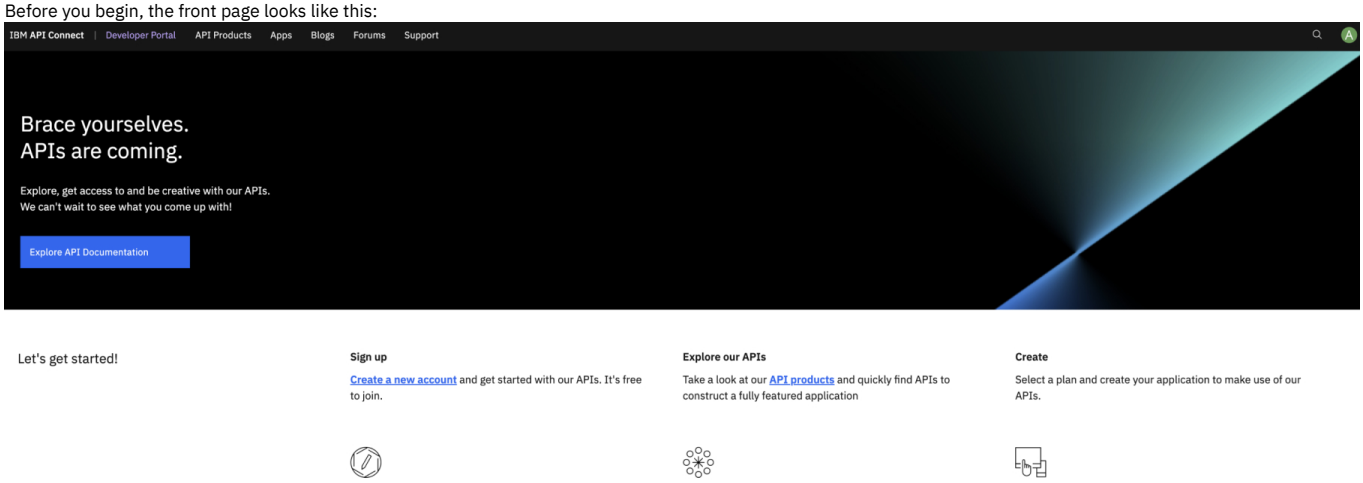
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

In this task, you create a view, customize that view to show tweets, then embed that view on the front page of your Developer Portal site.

Before you begin, the front page looks like this:

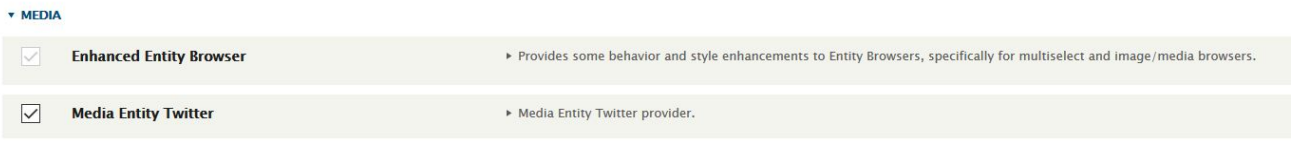


This tutorial takes you through the following steps:

1. Enable the Media entity Twitter module.
2. Create an entity called Twitter to store the tweets.
3. Add media items.
4. Create a carousel view.
5. Customize the carousel view.
6. Embed the carousel view on the front page of your Developer Portal.

Enable the `Media entity twitter` module

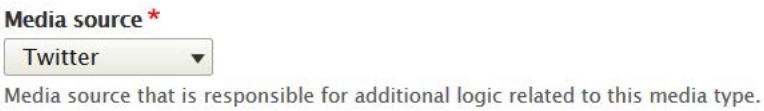
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend, enter media in the search bar.
4. Check Media Entity Twitter, under MEDIA.



5. Click Enable.

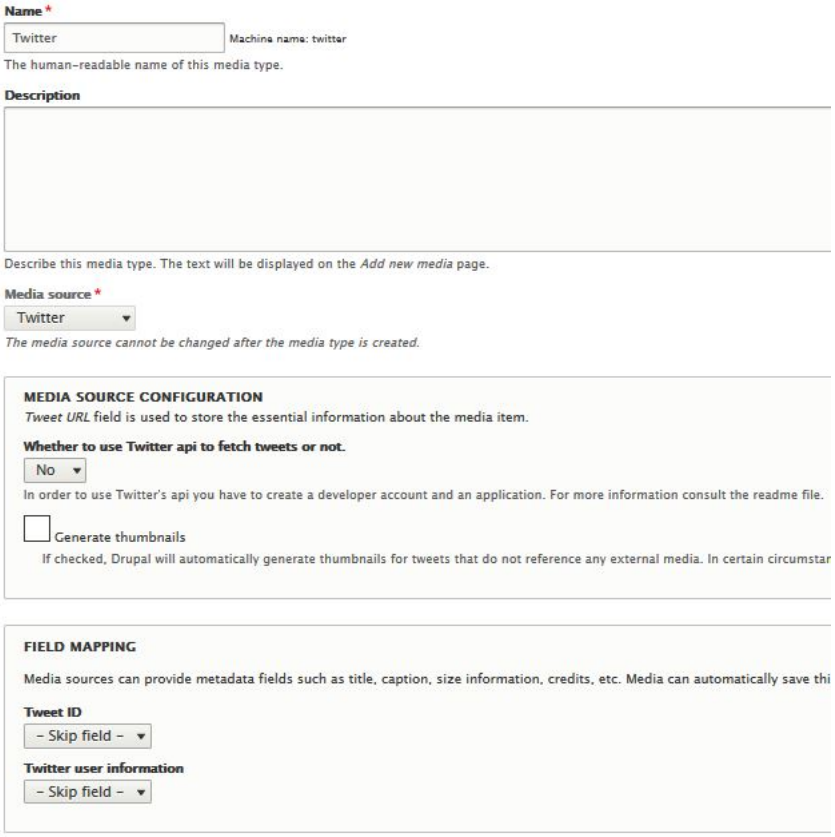
Create an entity called `Twitter` to store the tweets

1. Navigate to Structure > Media types > Add media type.
2. In the Name field, enter Twitter.

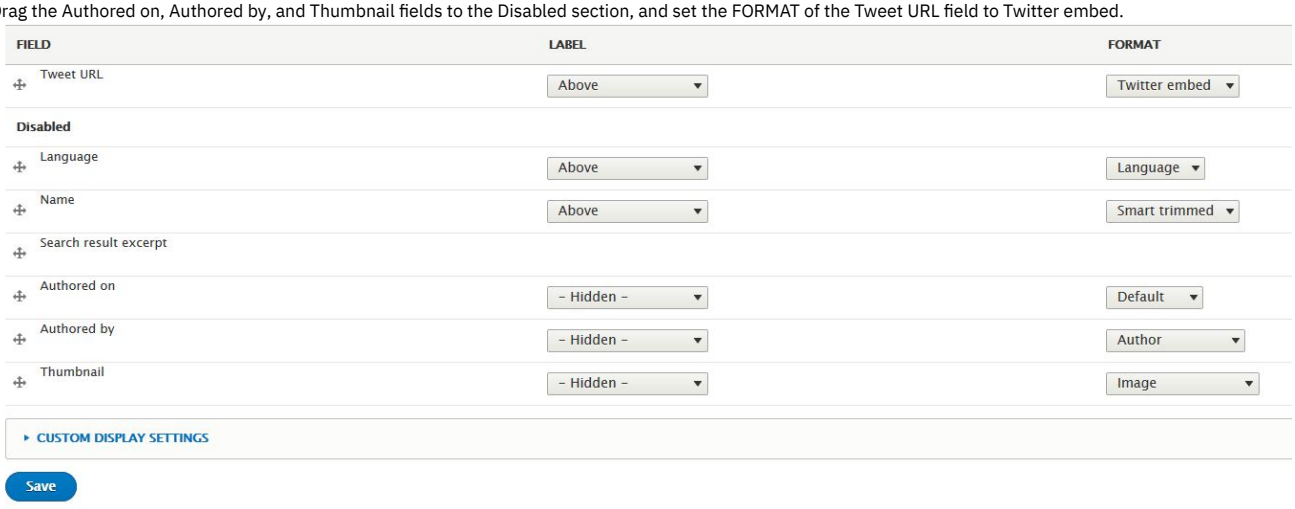


3. In the Media source drop-down, select Twitter.
4. Ensure that Whether to use Twitter api to fetch tweets or not value is set to No.

5. Ensure the values in the FIELD MAPPING section are set to - Skip field -.



6. Click Save.
7. Click Manage display from the Edit dropdown for the new `Twitter` media type.
8. Drag the Authored on, Authored by, and Thumbnail fields to the Disabled section, and set the FORMAT of the Tweet URL field to Twitter embed.

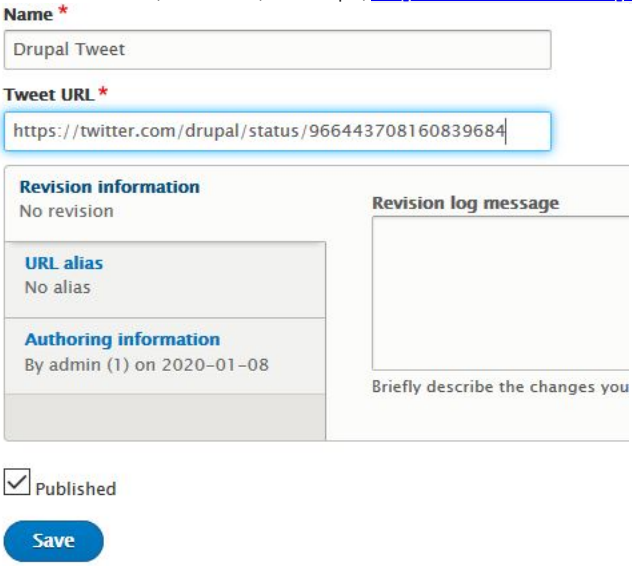


9. Click Save.

Add media items

1. Navigate to Content > Media > Add media > Twitter.
2. In the Name field, enter a name for the media, for example Drupal Tweet.

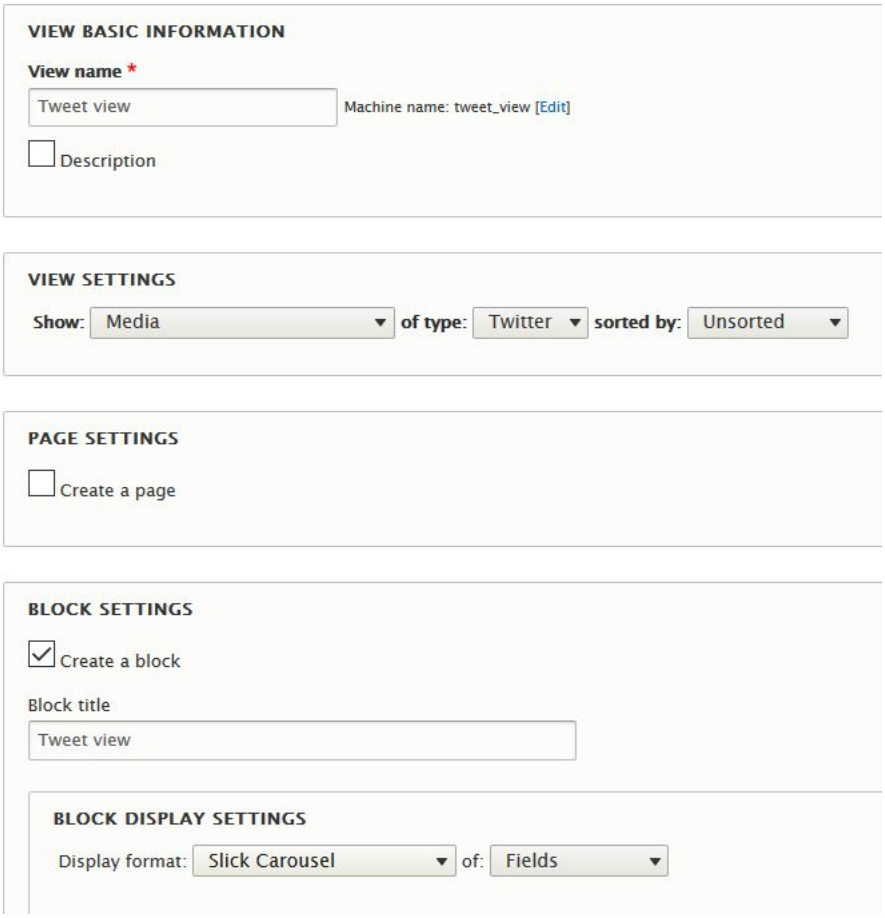
3. In the Tweet URL field, enter a URL, for example, https://twitter.com/drupal/status/966443708160839684.



4. Click Save.
5. Repeat these steps to add more tweets, you need at least 2 to make a carousel view.

Create a carousel view

1. Navigate to Structure > Views > Add view.
2. In the View name block, enter a name for your view, for example Tweet view.
3. Under VIEW SETTINGS, show Media of type Twitter sorted by Unsorted.
4. In the BLOCK SETTINGS, section check Create a block and ensure it is Display formatSlick Carousel of fields.

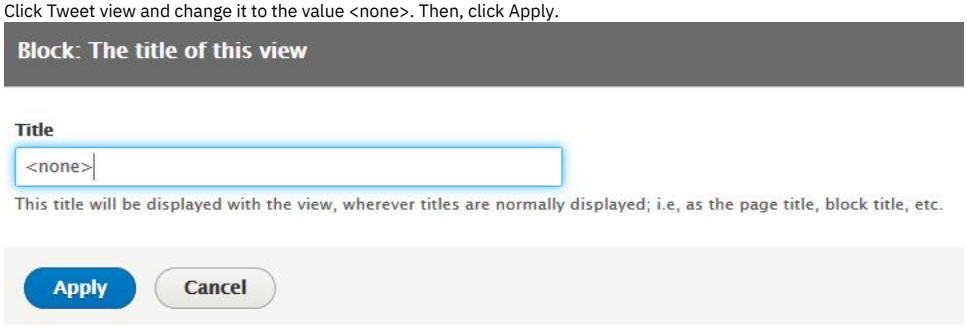


5. Click Save and edit.

Customize the carousel view

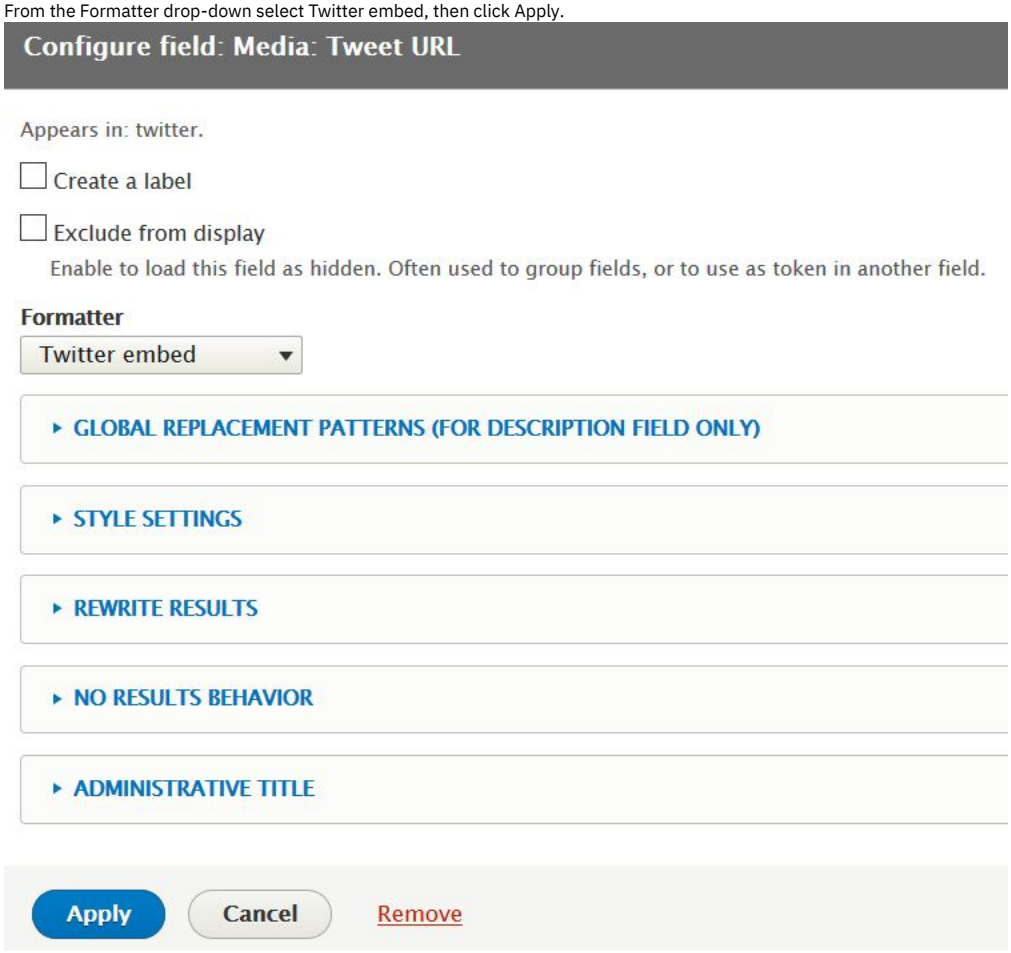
You are now on the page where you can edit your view.

1. If you are not the edit view page, navigate to Structure > Views, find the name of the view that you created, and click Edit.
2. Click Tweet view and change it to the value <none>. Then, click Apply.

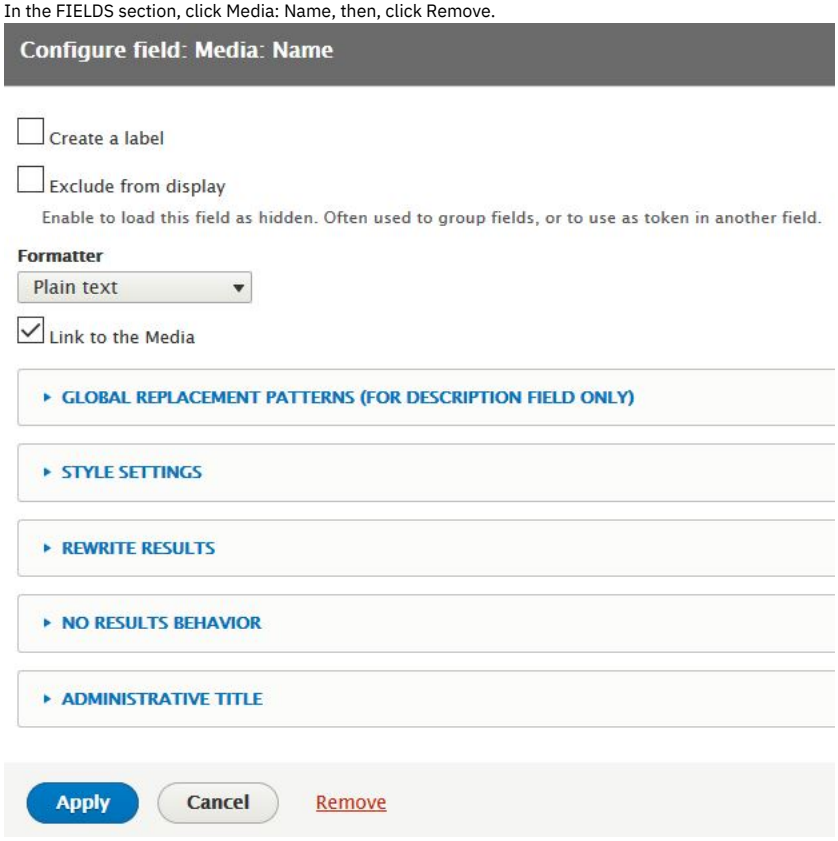


3. Click Add in the FIELDS section, then search for tweets, select the check box for Tweet URL. Then, click Add and configure fields.

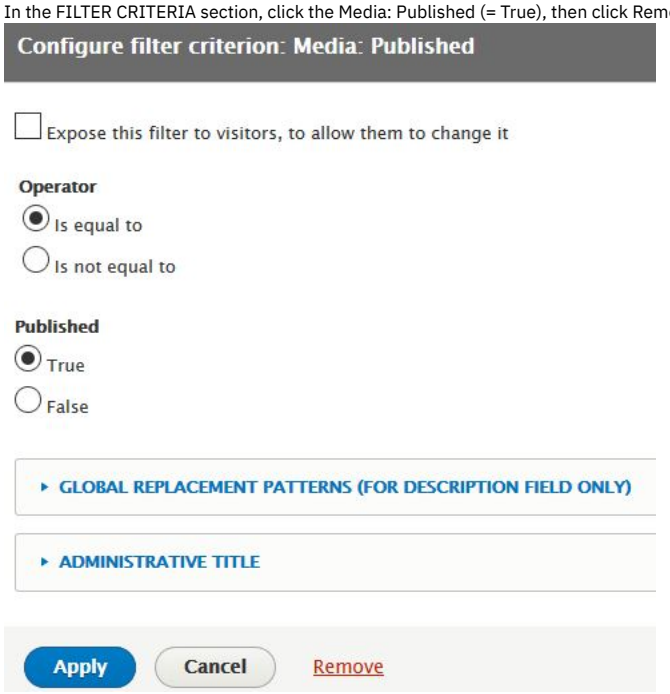
4. From the Formatter drop-down select Twitter embed, then click Apply.



5. In the FIELDS section, click Media: Name, then, click Remove.

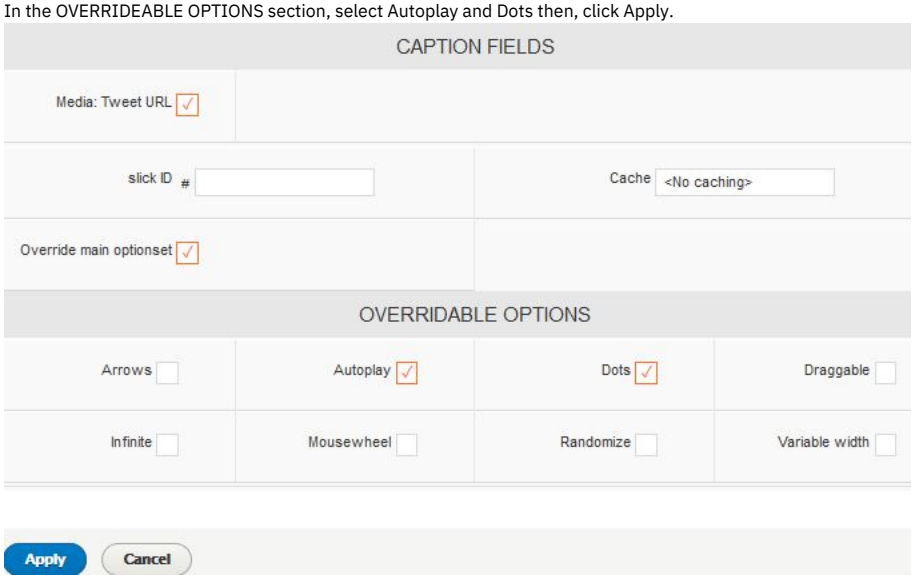


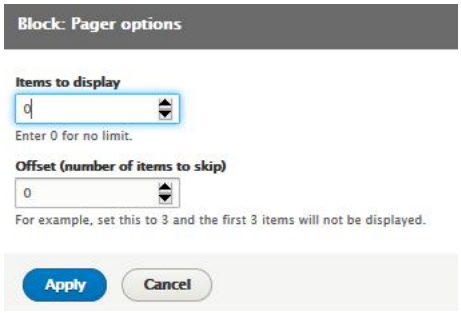
6. In the FILTER CRITERIA section, click the Media: Published (= True), then click Remove.



7. In the FORMAT section, click Settings next to Slick Carousel.
8. Set Skin Main to Default.
9. In the CAPTION FIELDS section, check Media: Tweet URL, and Override main optionset.

10. In the OVERRIDEABLE OPTIONS section, select Autoplay and Dots then, click Apply.

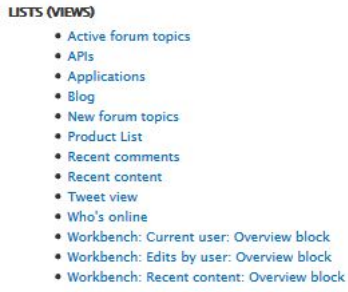




11. In the PAGER section, clickItems, set the field to 0 then, click Apply.
12. Click Save.

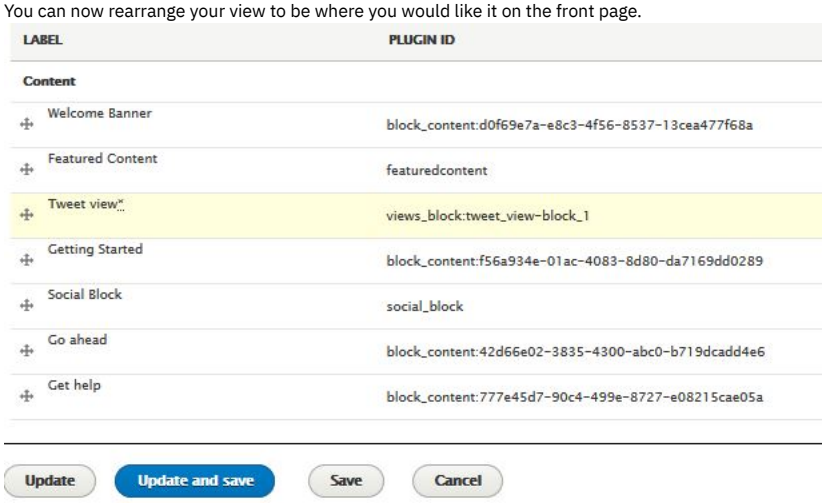
Embed the view on the front page of your Developer Portal

1. Navigate to Structure > Pages.
2. Click Edit on the welcome page.
3. Click Panels, then Content.
4. Click Add new block.



5. Select your new block, which is under LISTS(VIEWS).
6. Deselect Display title then, click Add block.

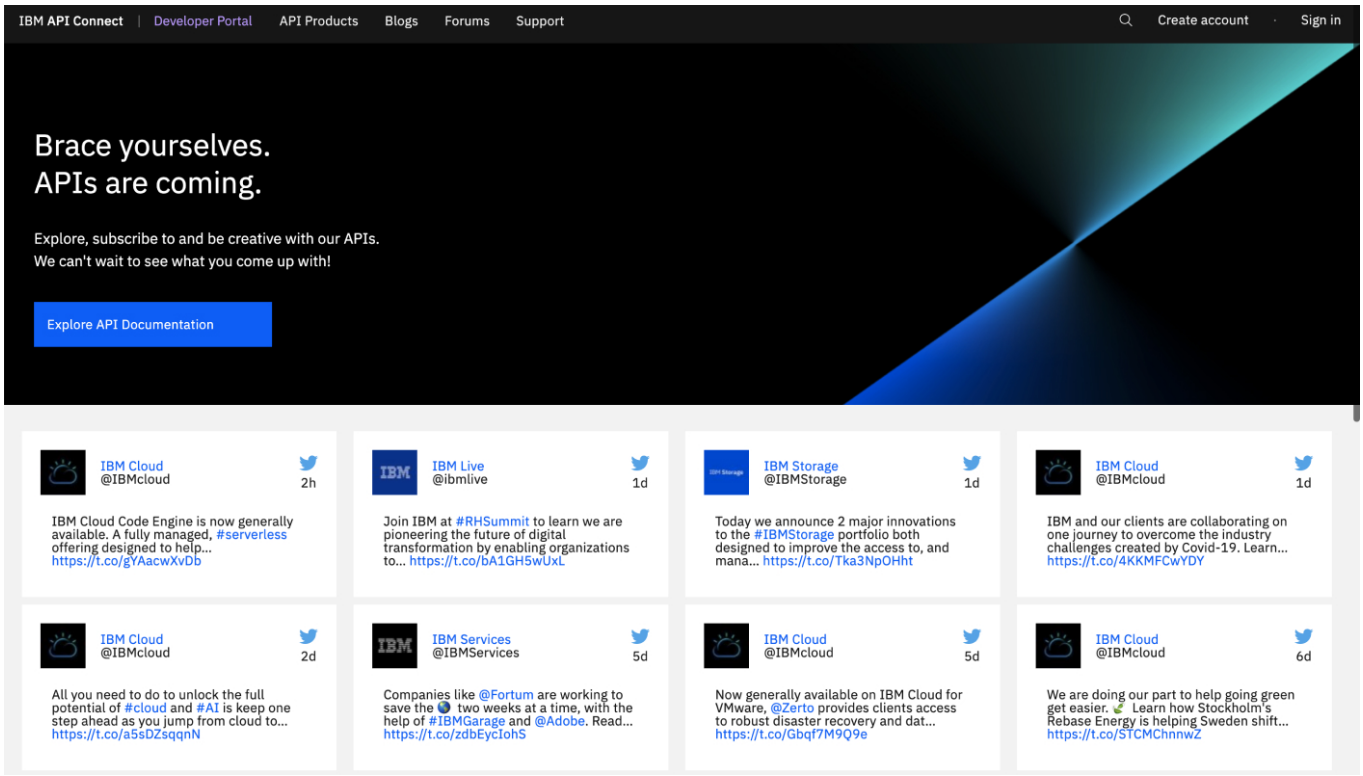
7. You can now rearrange your view to be where you would like it on the front page.



8. Click Update and save.

What you did in this tutorial

You can check that the view you created appears on the front page of your site page.



What to do next

You can edit your view at any time by navigating to Structure > Views and selecting your view by its name. You can also modify or delete the view's layout on the front page by navigating back to the layout section of your page under Structure > Pages.

You can also integrate your Twitter data dynamically. For more information, see the links below.

Related tasks

- **Integrating Twitter data into the social block**
- **Editing the social block**

Tutorial: Changing the front page of your Developer Portal

You can create and arrange content in your Developer Portal to provide a front page that aligns with your brand, and helps your users discover your APIs.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

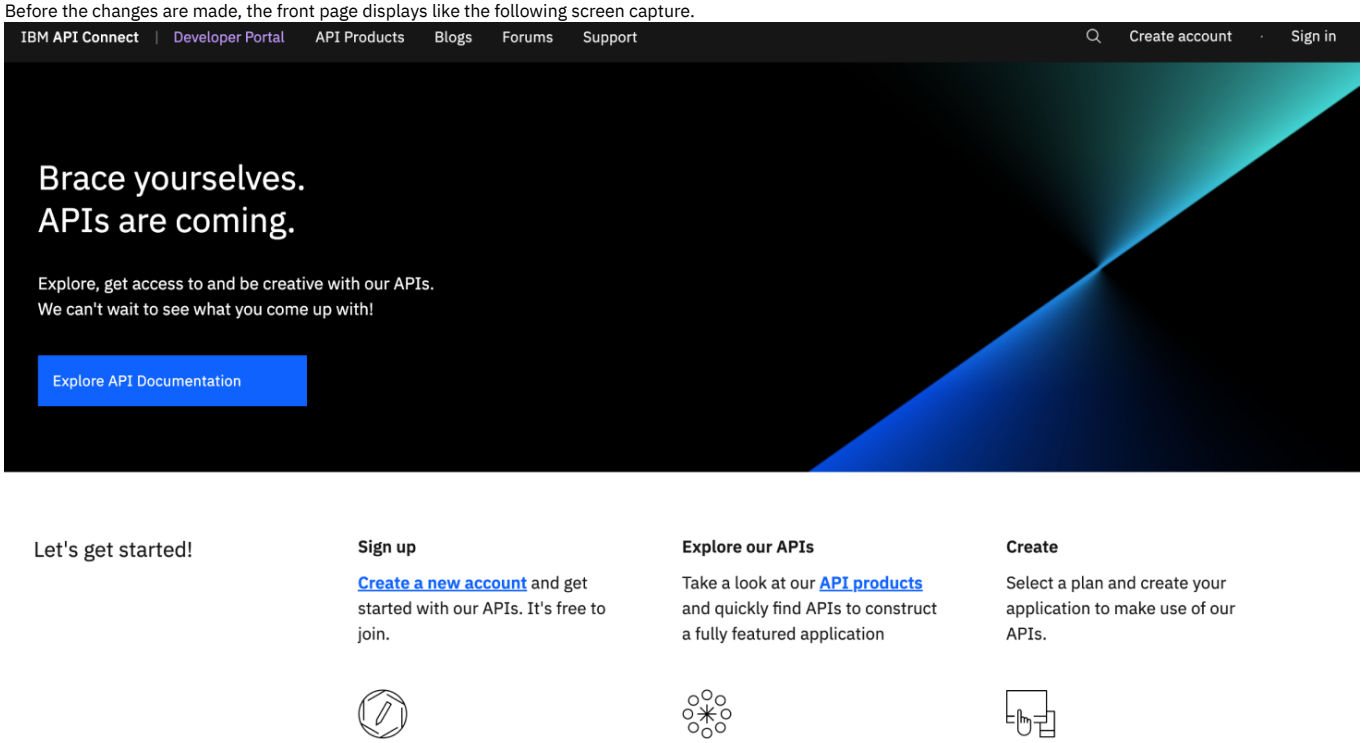
About this tutorial

In this tutorial, you create some content for a page, and then change the front page to be your new page.

This tutorial takes you through the following steps:

1. Creating some custom blocks.
2. Create and configure a page.
3. Adding the custom blocks to the page.
4. Changing the front page to be your new page.

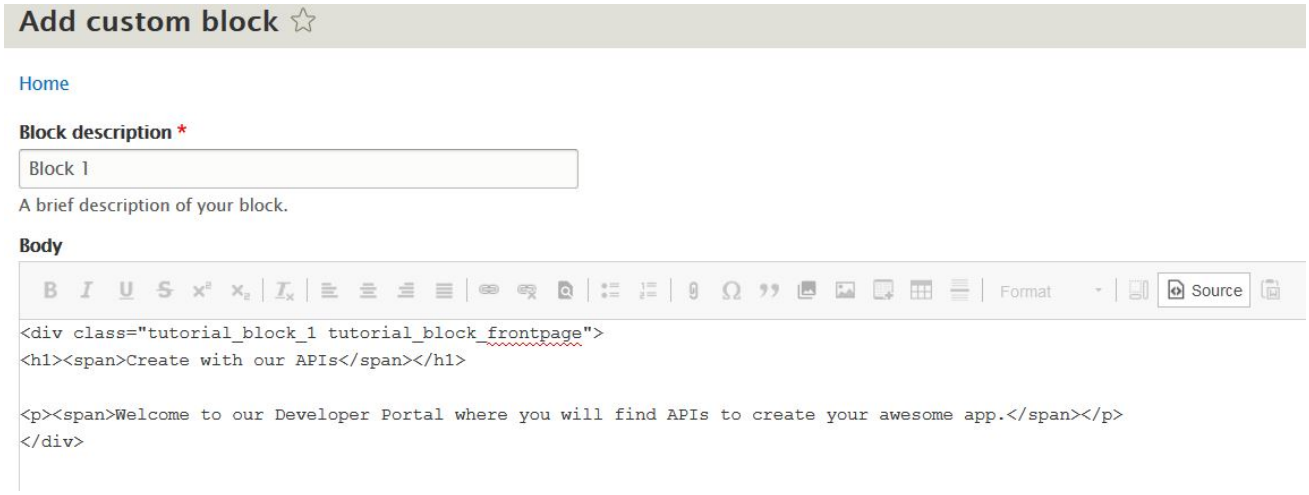
Before the changes are made, the front page displays like the following screen capture.



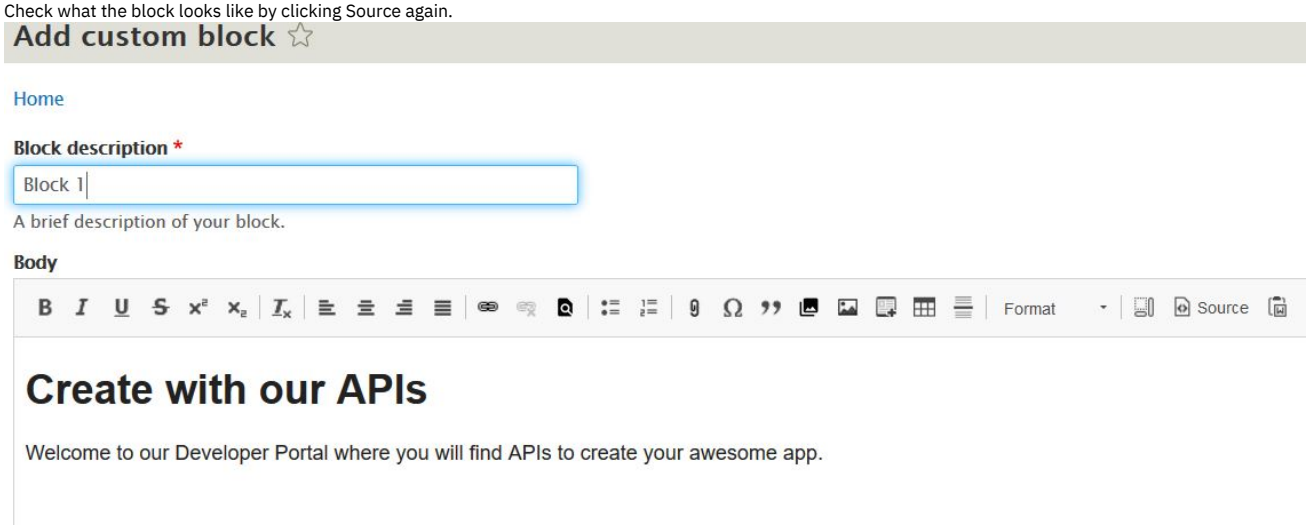
Creating some custom blocks

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Block layout > Add custom block
4. Enter `Block 1` in theBlock description.
5. In the body area, click Source, then enter the following HTML:

```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1><span>Create with our APIs</span></h1>

<p><span>Welcome to our Developer Portal where you will find APIs to create your awesome app.</span></p>
</div>
```



6. Check what the block looks like by clicking Source again.



Note: This block uses no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see Tutorial: Creating a custom theme for the Developer Portal.

7. Click Save.
You have created `Block 1`, repeat steps 3 to 7 to create `Block 2`, `Block 3`, and `Block 4`.

8. Enter `Block 2` in the Block description. In the body area, click Source, then enter the following HTML:

```
<div class="tutorial_block_2 tutorial_block_frontpage">
<p class="text-align-center">Explore and subscribe to our APIs.<br />
See what you come up with!</p>

<p class="text-align-center"> </p>

<div class="bannerButtonRow text-align-center"><a class="button cta-btns--white__btn"
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_product">Explore API
Documentation</a></div>
</div>
```

9. Enter `Block 3` in the Block description. In the body area, click Source, then enter the following HTML:

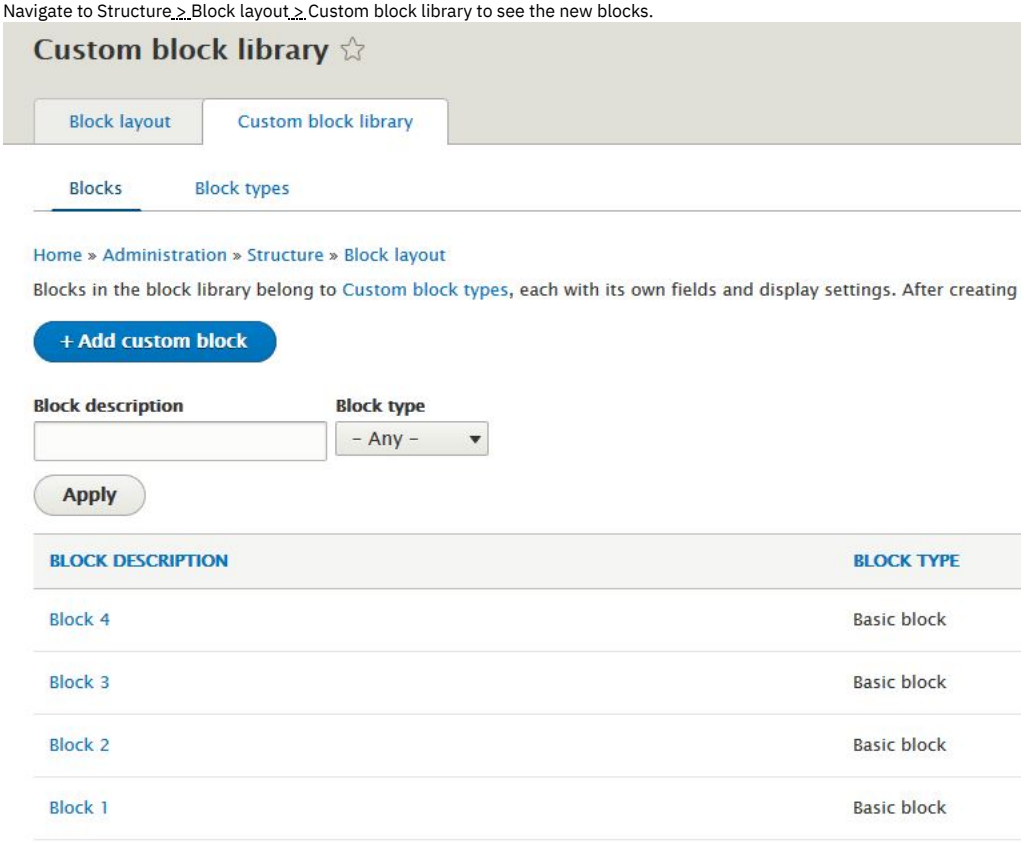
```
<div class="tutorial_block_3">
<p class="text-align-center"><a
href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_user_register">Create a new
account</a> and get started with our APIs. It's free to join.</p>
</div>
```

10. Enter `Block 4` in the Block description. In the body area, click Source, then enter the following HTML:

```
<div class="block-get-help">
<div class="get_help">
<div class="column col1">
<h3>Getting Help</h3>

<div>Be sure to check out these extra help resources.</div>
</div>

<div class="column col2">
<h4>Get Started</h4>

<div>Get started with our APIs by creating an account and exploring the documentation to find what's right for you.</div>
```

```
<div><a href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_user_register"><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpag
e_Layer_1" space="preserve" version="1.1" viewbox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Create an Account</a></div>
</div>

<div class="column col3">
<h4>Forum</h4>

<div>Ask a question in the forums or search to forum history to see if it's been asked before.</div>

<div><a href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_forum"><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpag
e_Layer_1" space="preserve" version="1.1" viewbox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Join the discussion</a></div>
</div>

<div class="column col4">
<h4>Contact Us</h4>

<div>Can't find the answer to your question? Need more help? Have some feedback? Let us know!</div>

<div><a href="#_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_contact"><svg
height="24"
id="_home_markdown_jenkins_workspace_Transform_in_SSMNED_v10cd_com.ibm.apic.devportal.doc_tutorial_portal_example_frontpag
e_Layer_1" space="preserve" version="1.1" viewbox="0 0 32 32" width="24" x="0px" xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" y="0px"> <g> <polygon points="15.293,10.707 19.586,15 8,15 8,17 19.586,17 15.293,21.293
16.707,22.707 23.414,16 16.707,9.293"></polygon> <path d="M16,2C8.269,2,2,8.269,2,16s6.269,14,14,14c7.731,0,14-6.269,14-
14S23.731,2,16,2z M16,28C9.383,28,4,22.617,4,16S9.383,4,16,4c6.617,0,12,5.383,12,12S22.617,28,16,28z"></path> </g>
</svg>Email us</a></div>
</div>
</div>
</div>
```

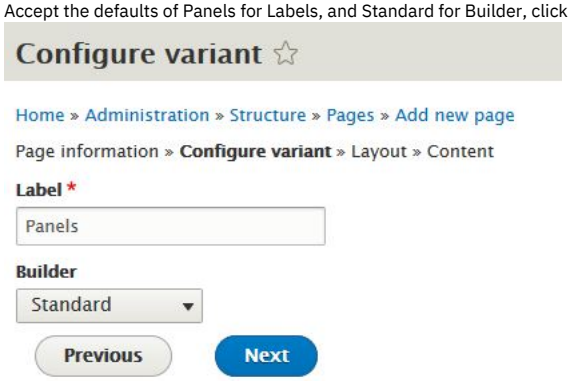
11. Navigate to Structure > Block layout > Custom block library to see the new blocks.



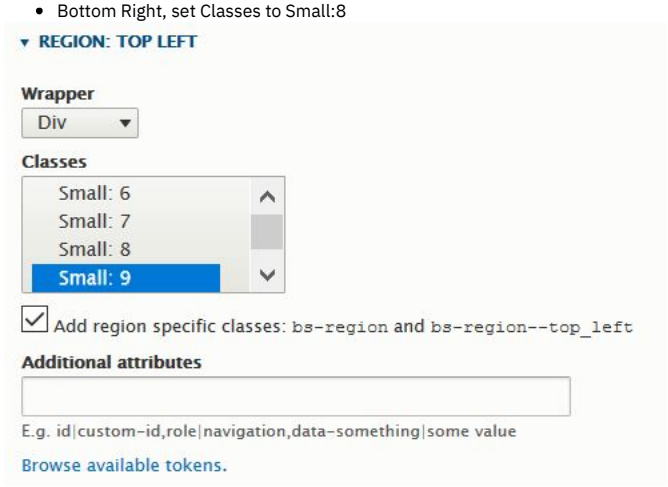
Create and configure a page

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages, clickAdd page.

3. Enter `Portal home` in the Administrative title field.
4. Enter `newhome` in the Path field.
5. Select **`Panels`** in the Variant title drop-down.
6. Leave other fields clear.
7. Click Next.
8. Accept the defaults of Panels for Labels, and Standard for Builder, click Next.



9. Select Two column bricks from Layout drop-down. Click Next.
10. Keep all settings except for the following:
    - Top Left, set Classes to Small:9
    - Top Right, set Classes to Small:3
    - Bottom Left, set Classes to Small:4
    - Bottom Right, set Classes to Small:8

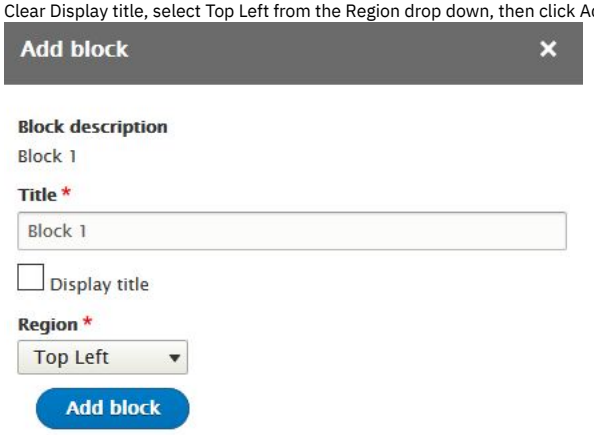


11. Click Next.

Adding the custom blocks to the page

1. Leaving Page title blank, click Add new block.
2. Select Block 1 from the Custom block list.

3. Clear Display title, select Top Left from the Region drop down, then click Add block.



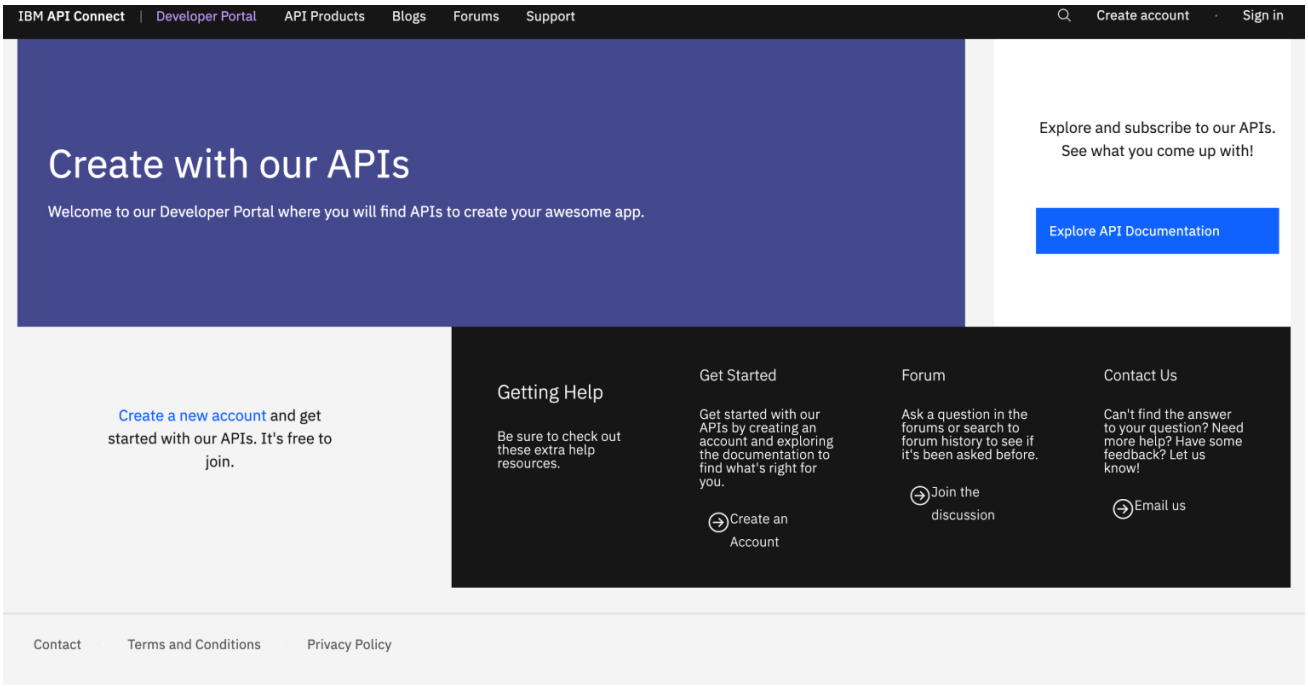
You added Block 1, repeat steps 1 to 3 to add Block 2, Block 3, and Block 4.

4. For Block 2 select region Top Right.
5. For Block 3 select region Bottom Left.
6. For Block 4 select region Bottom Right.
7. ClickFinish.
8. Click Update and save.

Changing the front page to be your new page

Note: In this tutorial, you create a new page and swap it with the default. If errors are made, the default page is still available as a backup.

1. Navigate to Structure > Pages.
2. Click the newhome URL link for your new Portal home page, check that your page looks as designed.



3. Navigate to Configuration > System > Basic site settings
4. In Front Page region set Default front page field tonewhome .
5. Click Save configuratoin.

What you did in this tutorial

The front page of your Developer Portal is now changed and laid out with the blocks of content that you specified.

What to do next

You can add further branding by adding content to your custom blocks such as images, or try to use one of the provided custom blocks such as the Featured Content Block. By default this block is English, you might want to consider adding versions in other languages.

Tutorial: Grouping products by category

You can configure the Developer Portal so that products are displayed in groups according to the category that they are tagged with.

You can use the **`Views`** module to fetch content from your Developer Portal site, such as lists of products, and present the content to users in different formats. You can combine multiple views on a new page to display whatever content you want.

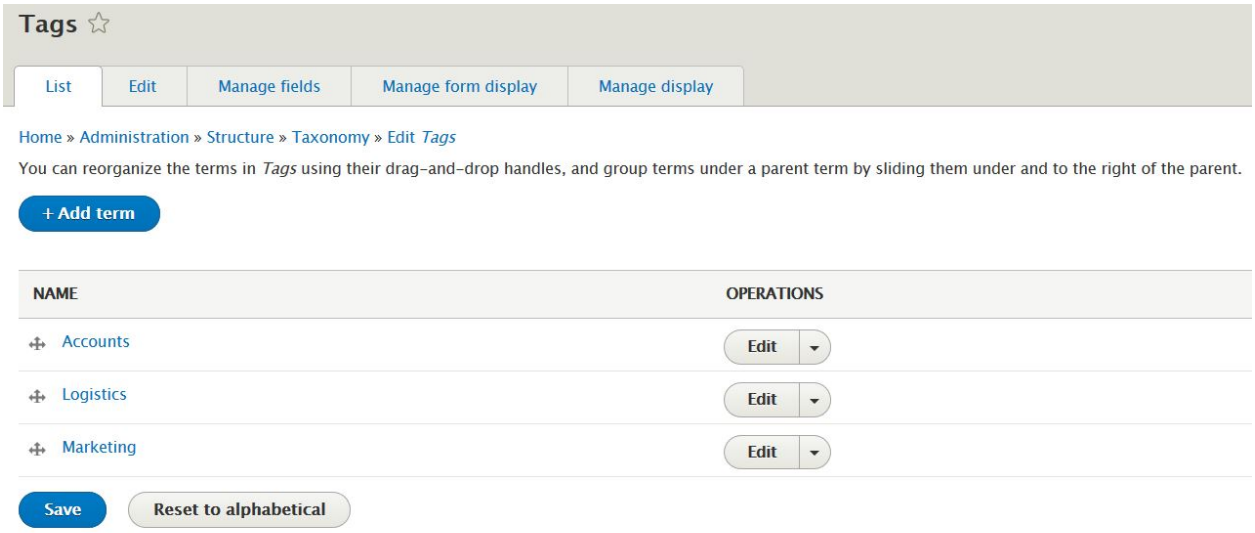
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

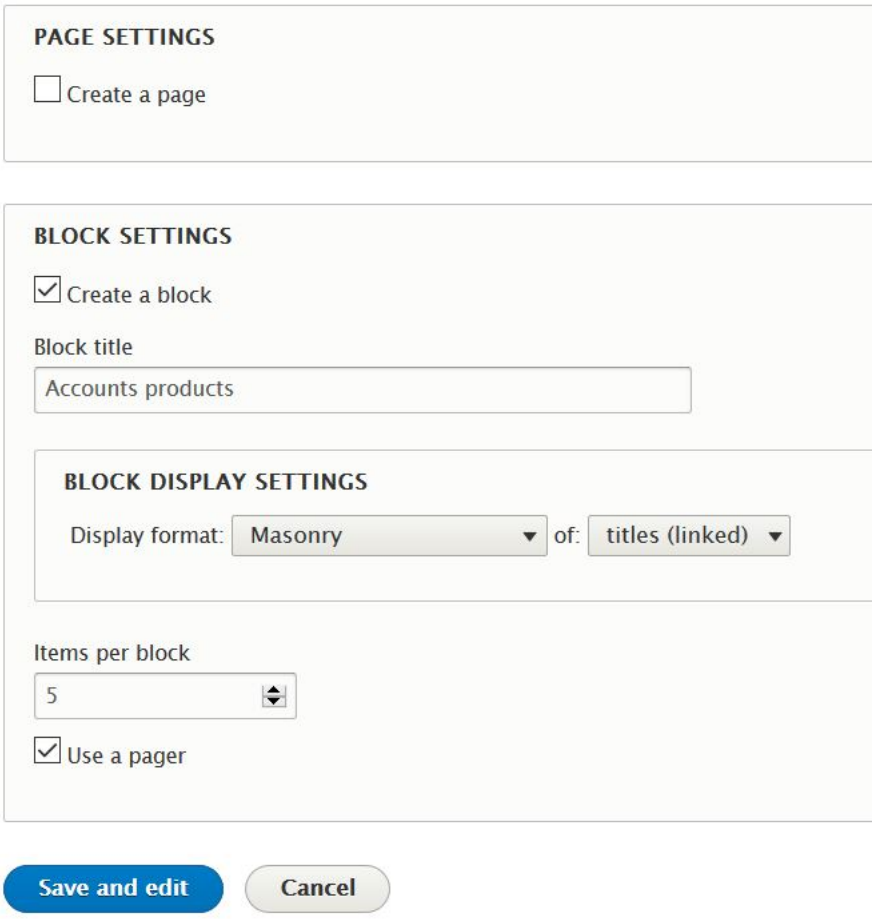
About this tutorial

In this tutorial you create three views in the Developer Portal, each one is configured to display a list of products according to their category. You then create a new page to show the views, where you see products that are grouped by category.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Create three new taxonomy tags.
    a. Navigate to Structure > Taxonomy > Tags.
    b. Click **`Add term`**.
    c. Enter `Accounts` in the **`Name`** field and click **`Save`**.
    d. Enter `Logistics` in the **`Name`** field and click **`Save`**.
    e. Enter `Marketing` in the **`Name`** field and click **`Save`**.
    f. Navigate to Structure > Taxonomy > Tags to show the list of the new tags.

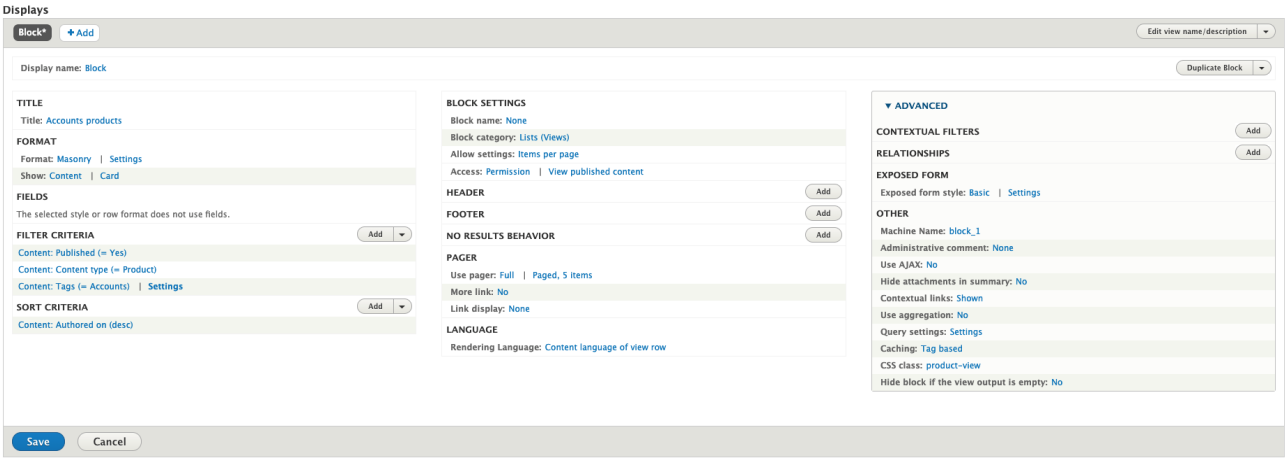


4. Create a view to display products with the **`Accounts`** category.
   a. Navigate to Structure > Views > Add new view.
   b. Enter `Accounts products` in the **`View name`** field.
   c. In **`View Settings`**, select **`Show`**: **`Content`**, **`of type`**: **`Product`**, **`sorted by`**: **`Newest first`**, from the drop-down lists.
   d. In **`Page settings`** leave the **`Create a page`** cleared.
   e. In **`Block settings`** select **`Create a block`** and enter `Accounts products` in the **`Block title`**.
   f. Ensure **`Block Display Settings`** has **`Display format`**: **`Masonry`**, **`of`**: **`tiles (linked)`**, and select **`Use a pager`**.

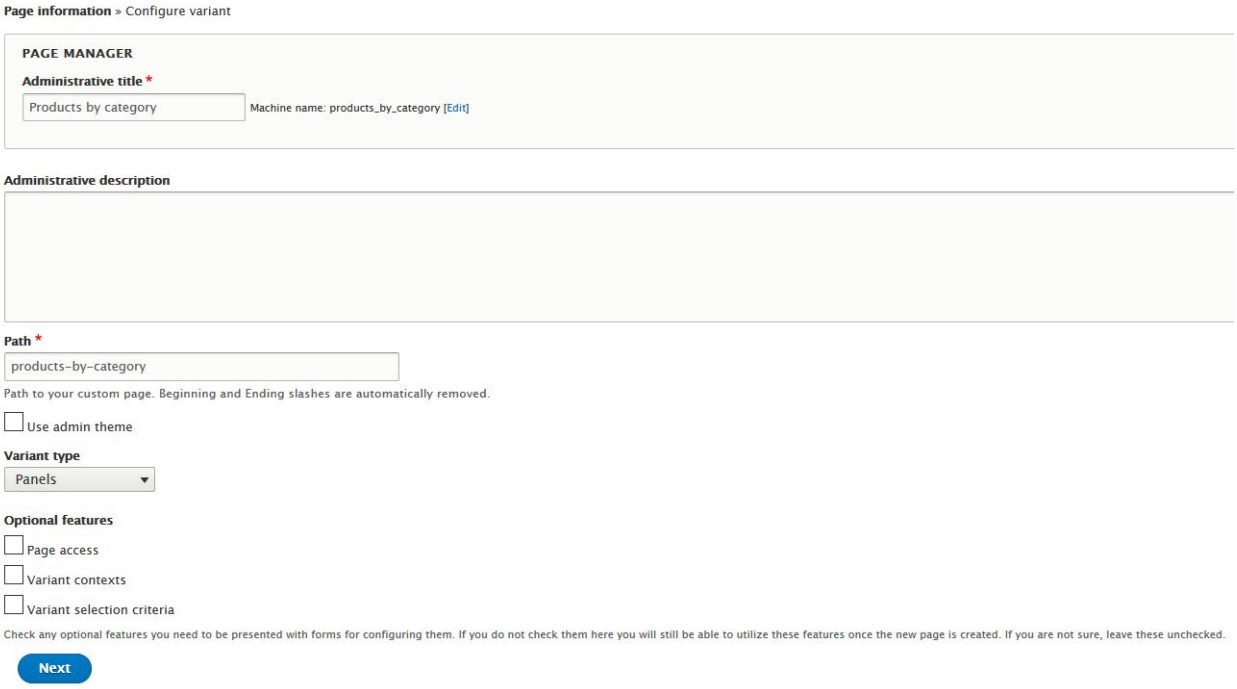


   g. Click **`Save and edit`**.
5. Configure the view block settings.
   a. Under the **`Format`** section click the **`Fields`** link next to the **`Show`** label.
   b. In the dialog box, select the **`Content`** radio button then click **`Apply`**.
   c. Select **`Card`** from the **`View mode`** drop-down list, then click **`Apply`**.
   d. Click **`Add`** next to the **`Filter Criteria`** section.

e. In the dialog box, search for `tag` then select the option **`Tags (apic_tags)`** from the list then click **`Add and configure filter criteria`**.
f. In the next dialog box, select the **`Tags`** radio button and the **`Client-side hierarchical`** then click **`Apply and continue`**.
g. Next set the **`Operator`** to **`Is all of`**, and from the **`Select terms from vocabulary Tags`** drop-down select **`Accounts`** and click **`Apply`**.
h. Under the **`Advanced`** options, click the **`None`** link next to the **`CSS class`** label.
i. If you are using the default API connect theme, then enter the CSS class name **`product-view`** and click **`Apply`**. If you have your own css class that you want to use, then enter its name in here instead.
j. Click **`Save`** to save all the changes.

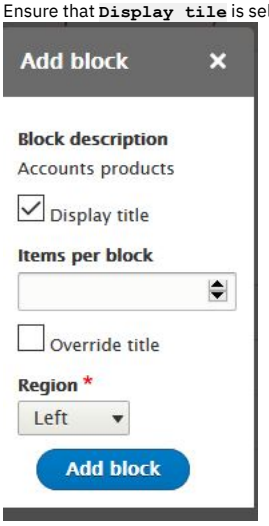


6. Create a view for the Logistics products.
   a. Select Structure > Views.
   b. Navigate to the **`Accounts products`** view and select the **`Duplicate`** option.
   c. Give the duplicated view the name **`Logistics products`** and click **`Duplicate`**.
   d. Click the Title **`Accounts products`** and change it to **`Logistics products`** and click **`Apply`**.
   e. Under the **`Filter Criteria`** section click the **`Content: Tags: (=Accounts)`** link.
   f. Change the value in the **`Select terms from vocabulary Tags`** drop-down to **`Logistics`** and click **`Apply`**.
   g. Click **`Save`** to save all the changes.
7. Repeat step 6 for Marketing products.
8. Next, create a page to display the views.
   a. Select Structure > Pages.
   b. Click **`Add page`**.
   c. Enter `Products by category` in the **`Administrative title`**, and `products-by-category` in the **`Path`**. Select the **`Panels`** option from the **`Variant type`** drop-down list then click **`Next`**.



   d. Change the **`Label`** value to `Products grouped by category` and set the **`Builder`** to **`Standard`** then click **`Next`**.
   e. From the **`Layout`** drop-down list select the value **`Three column`** then click **`Next`**.
   f. Leave all the defaults on the next page and click **`Next`**.
   g. Click **`+Add new block`**.
   h. Select **`Accounts products`** from the **`Lists (views)`** section.

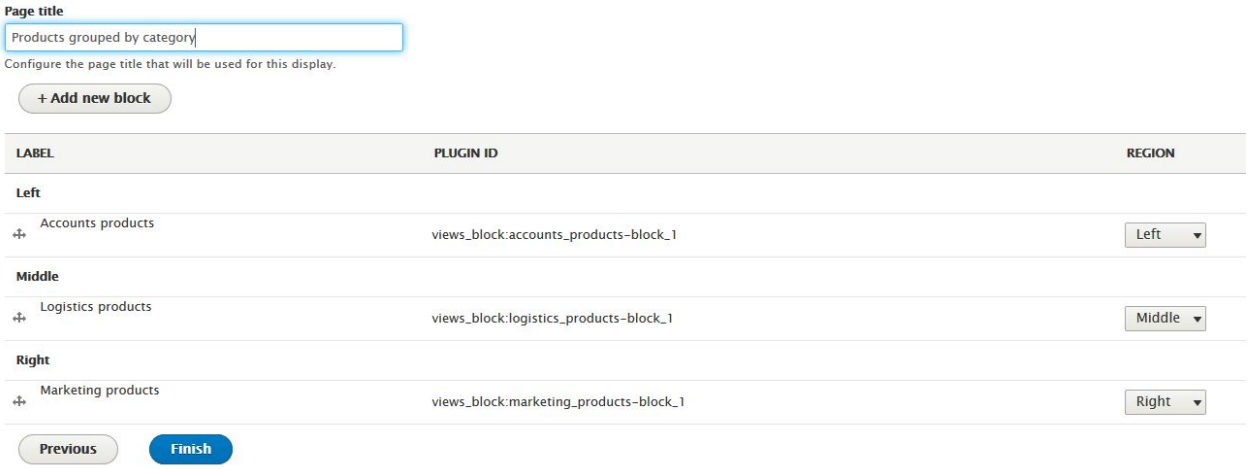
i. Ensure that **`Display tile`** is selected and **`Region`** is set to **`Left`**, click **`Add block`**.



j. Click **`Add new block`**. Select **`Logistics products`** view from the list and set the **`Region`** to **`Middle`**, click **`Add block`**.

k. Click **`Add new block`**. Select **`Marketing products`** view from the list and set the **`Region`** to **`Right`**, click **`Add block`**.

l. Enter `Products grouped by category` in the **`Page title`** field, and click **`Finish`**.



9. From your API Manager, create three new products, or update existing ones, to give each one a category value of either Accounts, Logistics, or Marketing. Publish each of the products to the catalog with the configured Developer Portal.
10. Check that the **`Products grouped by category`** page shows your published products by category.
    a. Select Structure > Pages.
    b. Click the **`/products-by-category`** path link.
    c. You are redirected to the **`/products-by-category`** page and can see the products that you assigned to these categories.

What you did in this tutorial

You created three new views in the Developer Portal, and configured each one to display a list of products according to their category. You then created a new page to show the views.

Tutorial: Adding a field to the sign up form

You might want to collect and store additional information about your Developer Portal users. For example, you might want to collect a department name, employee ID, or similar data when a user registers for an account on the portal.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

About this tutorial

In this tutorial, you will add a Department code field to the Sign up form. This field is required for all portal users.

This tutorial takes you through the following steps:

1. Adding a field to a user entity.
2. Adding the new field to the Sign up form.
3. Testing the new field.

Adding a field to a user entity

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > People > Account settings > Manage fields

You are on the **`Manage fields`** tab.

1. Click **`+ Create a new field`**.
2. In the **`Select a field type`** dropdown, select **`Text (plain)`**.
3. In the **`Label`** field enter, **`Department code`**.
4. Click **`Save and continue`**.

You are on the **`Field settings`** tab.

1. Accept the defaults of **`Maximum length`** (255) and **`Allowed number of values`** (Limited:1).
2. Click **`Save field settings`**.

You are on the **`Edit`** tab.

1. Add some help text. This is information that is displayed to the user to assist them completing the field on a form. Examples or links to further information are useful here. We are going to add an example in this tutorial.
2. Add **`Example department code: DEP123.`** into the **`Help text`** field.
3. Check the **`Required field`**.
4. Click **`Save settings`**.

The field has now been created. You are back on the **`Manage fields`** tab.

1. Check that **`Department code`** is a listed field.

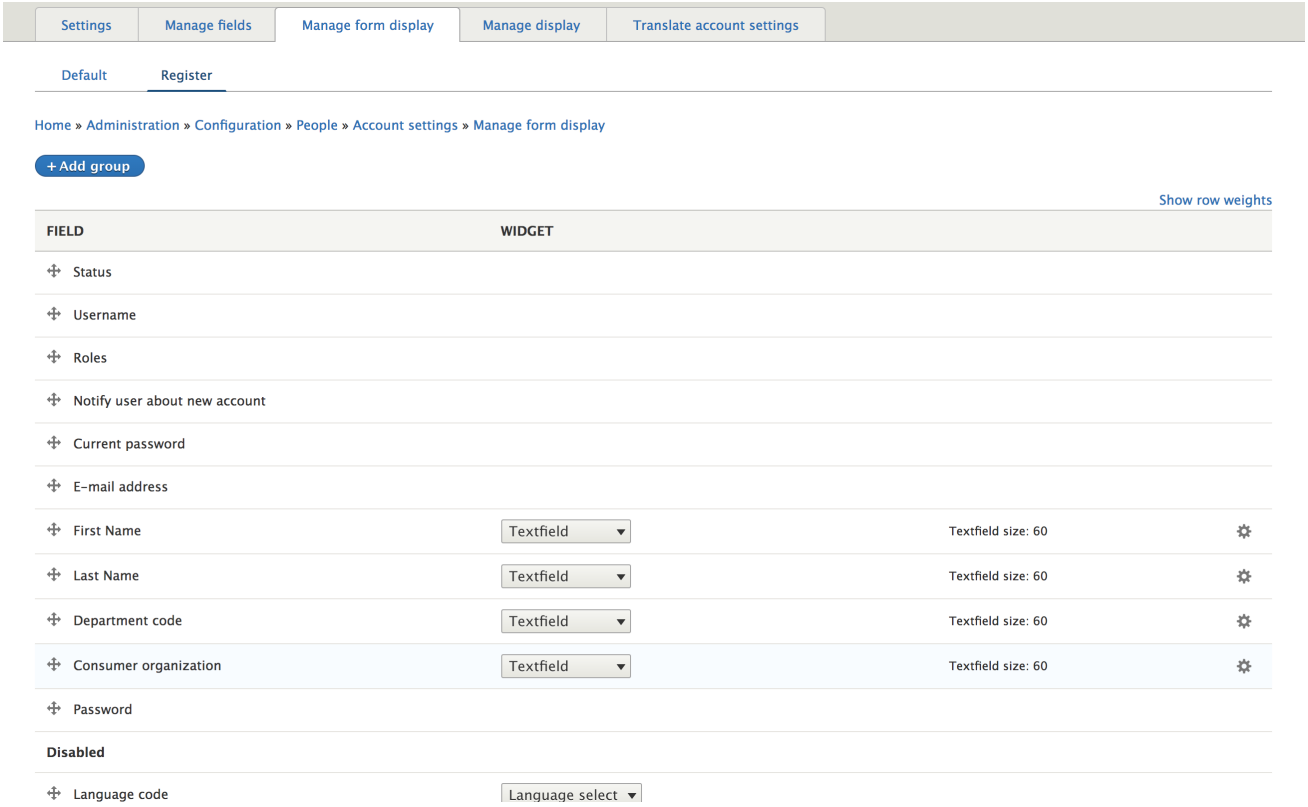
Adding the new field to the Sign up form

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Configuration > People > Account settings > Manage form display

You are on the *Manage form display* tab. These tabs, one for each form, show the fields that are displayed from the user entity, and in which order.

1. Switch to the **`Register`** tab. Notice that initially the new **`Department code`** field is underneath the **`Disabled row`** in the page. This means that it isn't shown on the form.

2. Click the Department code icon and drag it so that it is between Last name and Consumer organization in the list.



3. Click Save.

Testing the new field

1. Sign out as the Admin user.
2. Click **`Create account`**.
3. Observe that the **`Department code`** field is now part of the sign up form.

What you did in this tutorial

You added a Department code field to the Sign up form, then tested the form to ensure that the **`Department code`** field is now part of the sign up form.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this you would include the configuration in your custom module. For more information, see this Drupal documentation.

You might want to add extra validation to the Department code field on the Sign up form. For more information, see Tutorial: Adding validation to a field on the sign-up form.

Tutorial: Adding validation to a field on the sign-up form

You might want to add extra validation to a field to help your Developer Portal users. For example, you might want to validate a department name, employee ID, or similar data when a user registers for an account on the portal.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. You must have completed the tutorial about Adding a field to the sign up form, where you added a Department code field to the Sign-up form.

About this tutorial

In this tutorial, you add some validation to the Department code field on the Sign-up form. This is a required field for all portal users and is a string of the format **`DEPnnn`**, where **`DEP`** is the string prefix and **`nnn`** is a three-digit number, for example, **`DEP123`**. The validation is done by using a custom module that is already available in IBM APIC Portal - addons. However, for more information, see Custom module development: creating a module skeleton.

This tutorial takes you through the following steps:

1. Cloning and packaging the validation module.
2. Installing and enabling the validation module.
3. Testing the validation in the sign up form.

Look at the files we will use in this link IBM APIC Portal - addons.

- user_field_example.info.yml - this is the metadata for the module, defining things like the name and version, and containing other important information that is required by Drupal to import and enable the module.

```
name: 'IBM APIC Portal - user field example'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of validating a custom user field'
package: 'Custom'
core_version_requirement: ^8 || ^9 || ^10
version: 1.0.0
project: 'user_field_example'
dependencies:
  - ibm_apim
  - auth_apic
```

- user_field_example.module - this contains a validation function and adds this to the Sign-up form by using the `hook_form_alter` function.

```
<?php
use Drupal\Core\Form\FormStateInterface;

/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
  if ($form_id === 'user_register_form' ) {
      $form['#validate'][] = 'user_field_example_validate_department_code';
  }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 *   where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
  $dept_code = $form_state->getValue('field_department_code')[0]['value'];
  $valid = preg_match('/^DEP\d{3}$/', $dept_code);
  if (!$valid) {
    $form_state->setErrorByName('field_department_code', t('Invalid department code.'));
  }
}
```

Cloning and packaging the validation module

Validation is added to the **Department code** field on the Sign-up form by using a custom module.

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

   ```
   git clone https://github.com/ibm-apiconnect/devportal-addons
   ```

2. Package up the custom module.

   ```
   cd devportal-addons/apic_v10/modules
   tar -czf user_field_example.tgz user_field_example/
   ```

Installing and enabling the validation module

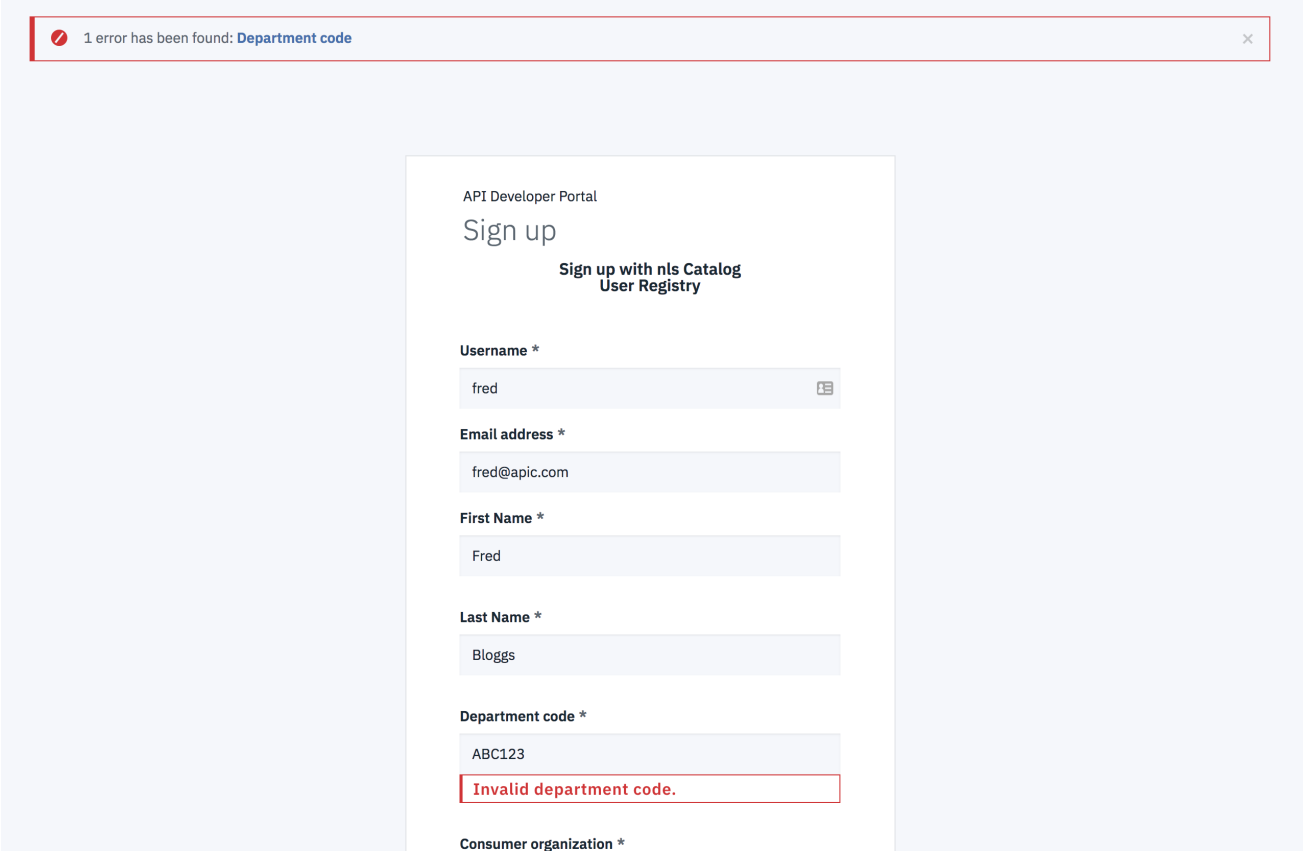
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend > Install new module.
4. Click **Browse file** from the **Upload a module or theme archive** section.
5. Browse to and select **user_field_example.tgz**, the packaged custom module that you created earlier. Click **Open**.
6. Click **Continue**.
7. Click the **Enable newly added modules** link.
8. Enter **IBM APIC Portal - user field example** in the filter box.
9. Check the checkbox next to **IBM APIC Portal - user field example**.
10. Click **Enable**.

The module is now installed and enabled.

Testing the validation in the sign-up form

1. Sign out as the Admin user.
2. Click **Create account**.
3. Complete all fields, set **Department code** to **ABC123**.
4. Click **Sign up**.

5. Observe the failure message and highlighting on the **`Department code`** field.



6. Change the **`Department code`** field to **`DEP245`**, reenter the password and click **`Sign up`** to see that a valid code is accepted.

What you did in this tutorial

You added some validation to the Department code field on the Sign-up form, then tested the form to ensure that the **`Department code`** field accepts only codes in the format of **`DEPnnn`**.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this availability, you would include the configuration in your custom module. For more information, see Drupal custom modules.

Tutorial: Adding a field group to group fields in a content type

You might want to change how your user account pages are displayed when you have lots of fields.

If you have many fields for one of your content types, for example, a user, then you might want to group some of these fields together when you display them to other users, or when these fields are being edited inside a user form.

Field groups allow you to create a parent field that contains other fields, set up as children, within that group. The **`Fieldgroup`** forms come with default HTML wrappers such as vertical tabs, horizontal tabs, accordions, fieldsets, or div wrappers.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

About this tutorial

You will create field groups for various groups of user information. These groups can then be seen under the users own section on the account view and edit pages.

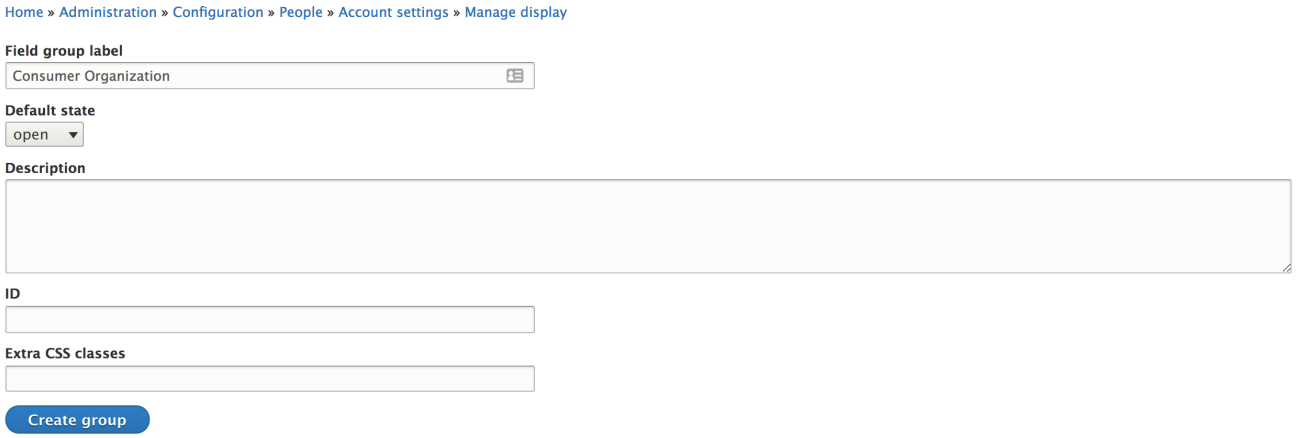
This tutorial takes you through the following steps:

1. Adding a field group called Consumer Organization to the user account display.
2. Adding a field group called Personal Details to the user form display.

Adding a field group called Consumer Organization to the user account display

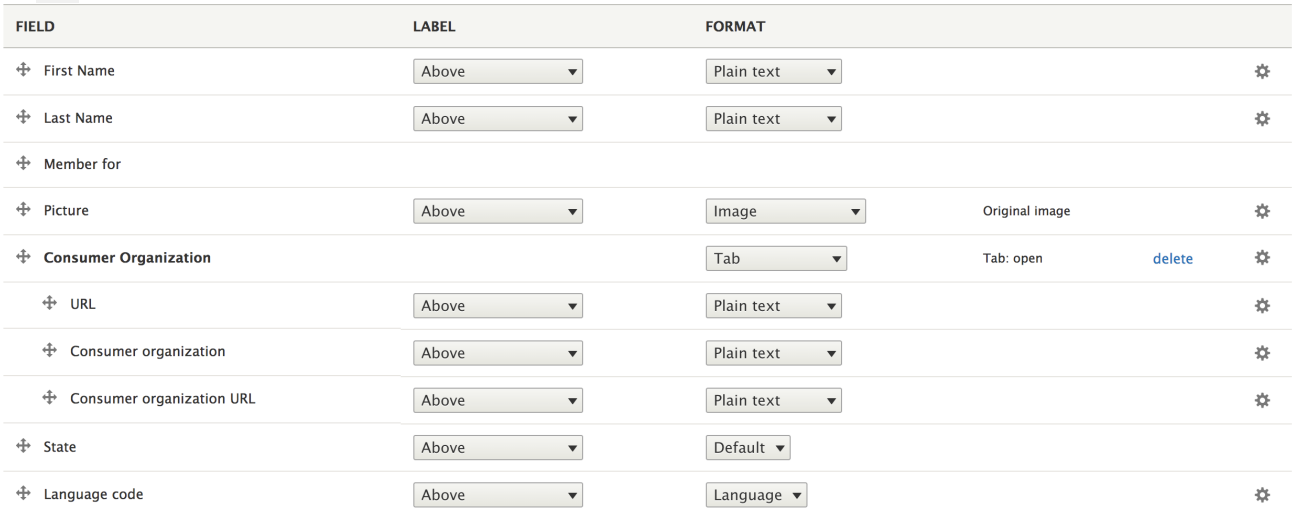
1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > People > Account settings > Manage display.
4. Click **`+ Add field group`**.

5. Select **tab** as your group type.
6. Enter **Consumer Organization** as the label.
7. Click **Save and continue**.
8. Set the default state to **open**. This sets the tab to be expanded by default.
9. Click **Create group**.

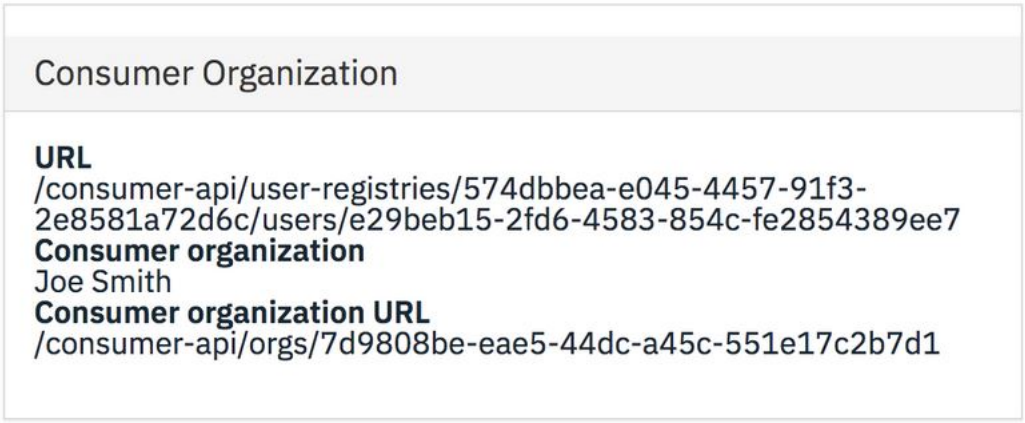


When the group is created, you can assign fields to that group and rearrange the group on the page. After you create the group, you are on the **Manage Display** page.

1. Drag your new group from the disable section up into the higher section, just underneath **Picture**.
2. Drag **URL**, **Consumer Organization**, and **Consumer Organization URL** underneath the **Consumer Organization** group row, and slightly to one side. It fixates itself underneath the parent, and this is now set as the child of that group.
3. Click **Save**.



You are now able to see the new group of fields on the account page of a user.



Adding a field group called Personal Details to the user form display

1. Navigate to Configuration > People > Account settings > Manage form display.

2. Click **Add field group**.
3. Select **tab** as your group type.
4. Enter **Personal Details** as the label.

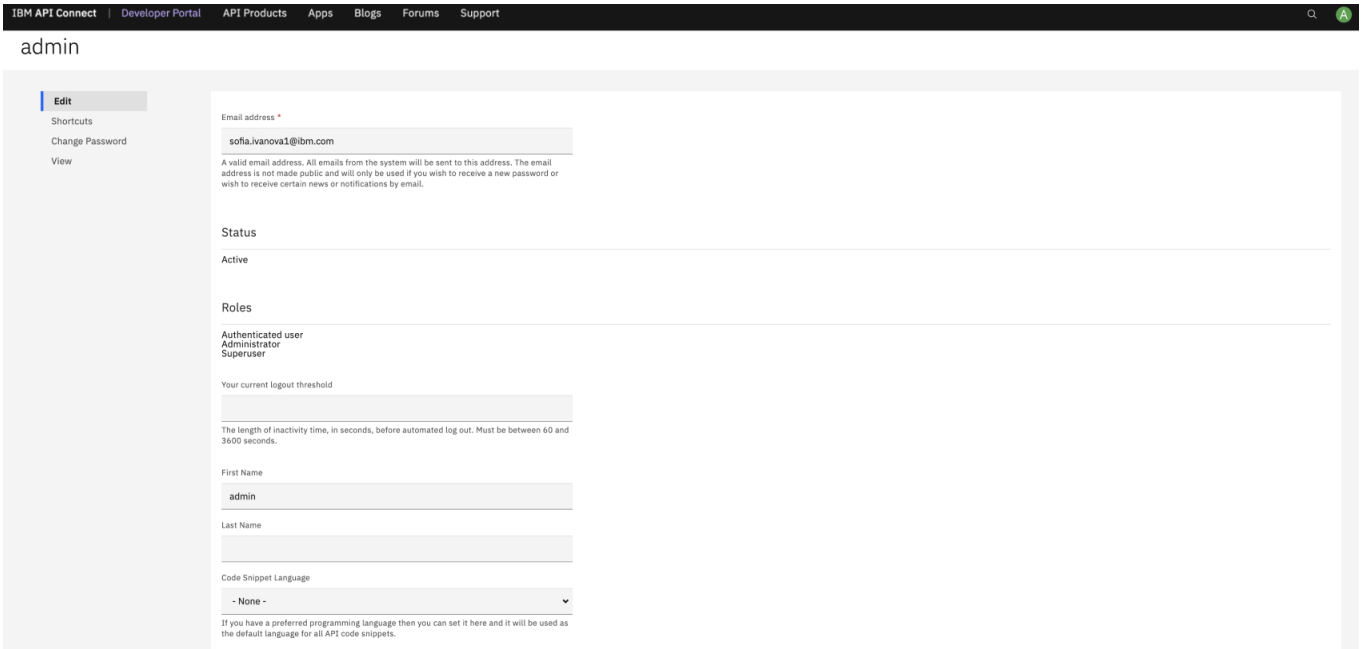
5. Click **Save and continue**.
6. Set the default state to **open**. This sets the tab to be expanded by default.
7. Click **Create group**.

Again, when the group is created, you can assign fields to that group and then rearrange the group on the page as you require. After you create the group, you are returned to the **Manage Display** page.

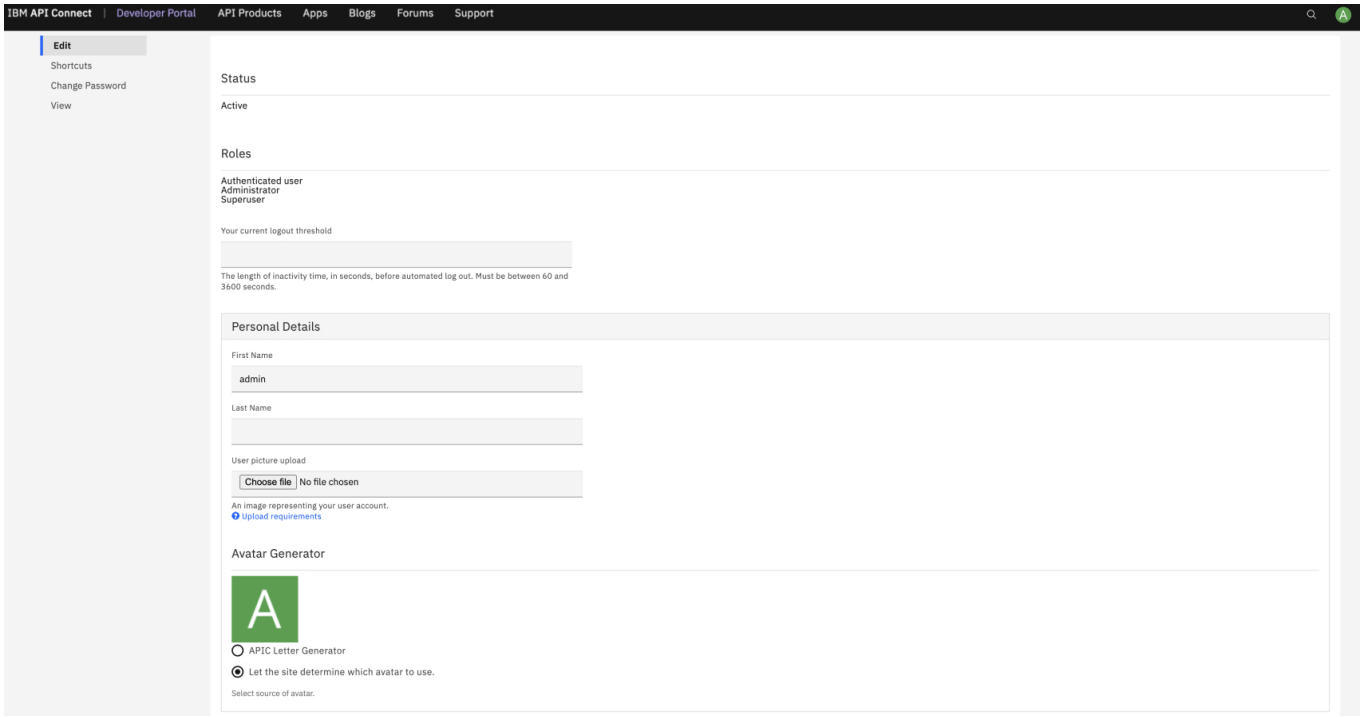
1. Drag your new group from the disabled section up into the higher section, onto the highest row.
2. Drag **First Name**, **Last Name**, **User picture upload**, and **Avatar Generator** to underneath the **Personal Details** group row. It fixates itself underneath the parent, and this is now set as the child of that group.
3. Click **Save**.

You and your user can now see the new grouped section when you edit their account.

Before:



After:



What you did in this tutorial

You have successfully created field groups for various groups of user information, and presented them to the user in its own section on the account view and edit pages.

You can check whether your field groups are applied, by viewing the profile of another user. Do this check by finding any other users published content and clicking the associated avatar. Or, from the administrator dashboard you can click People and then click a user. You can see the new Consumer Organization field group on their profile page. If you then click **`Edit`** for the user, you can see the Personal Details field group.

What to do next

You can edit the user displays anytime by navigating back to Configuration > People > Account settings > Manage display or Manage form display.

You can delete the field group by clicking the **`delete`** button in the field group row on either of these pages. You can also modify the type of group it is, for example, vertical tabs, horizontal tabs, accordions, and field sets.

Tutorial: Adding a custom block to the front page

You can customize the appearance of the front page of your Developer Portal by adding a custom block. Blocks are boxes of content that are rendered into an area, or region, of a web page such as the Products page or Apps page that can be displayed in regions, such as footer or sidebar, on your page.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so. The tutorial Adding a custom block to a page other than the front page explains how to add a custom block to any other page.

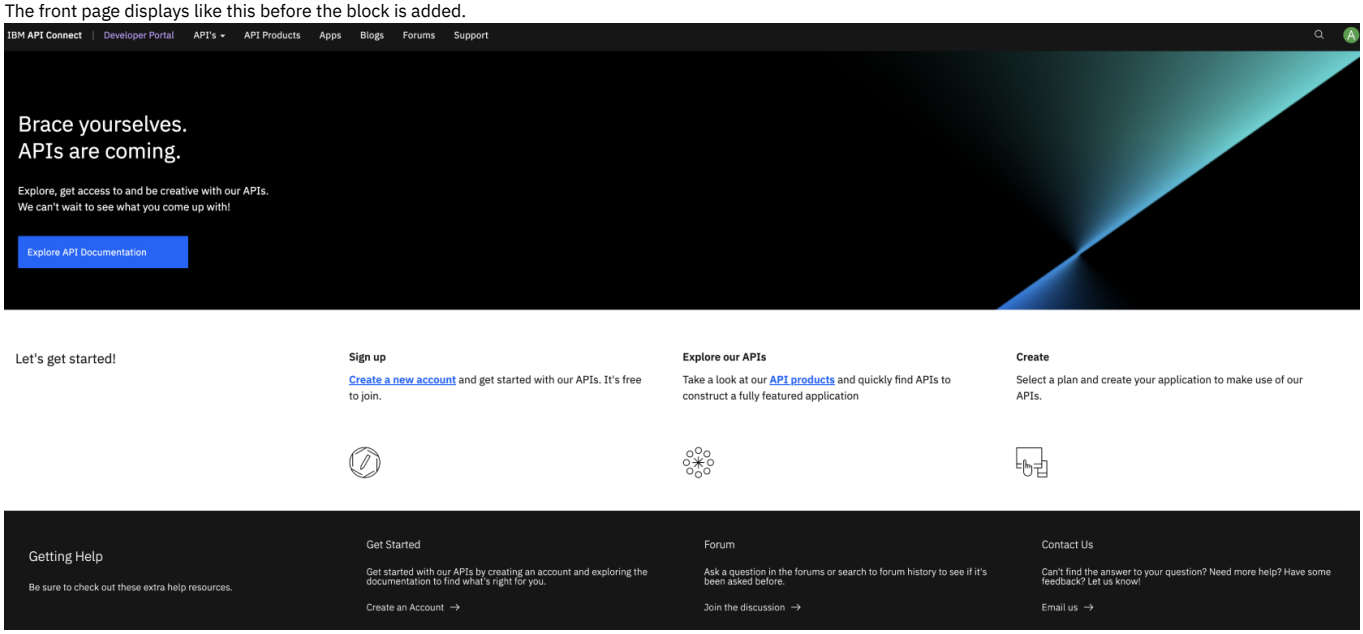
About this tutorial

In this tutorial, you create a block that contains a marketing announcement and add it to the front page of your Developer Portal.

This tutorial takes you through the following steps:

1. Creating a custom block.
2. Adding the custom block to the front page.

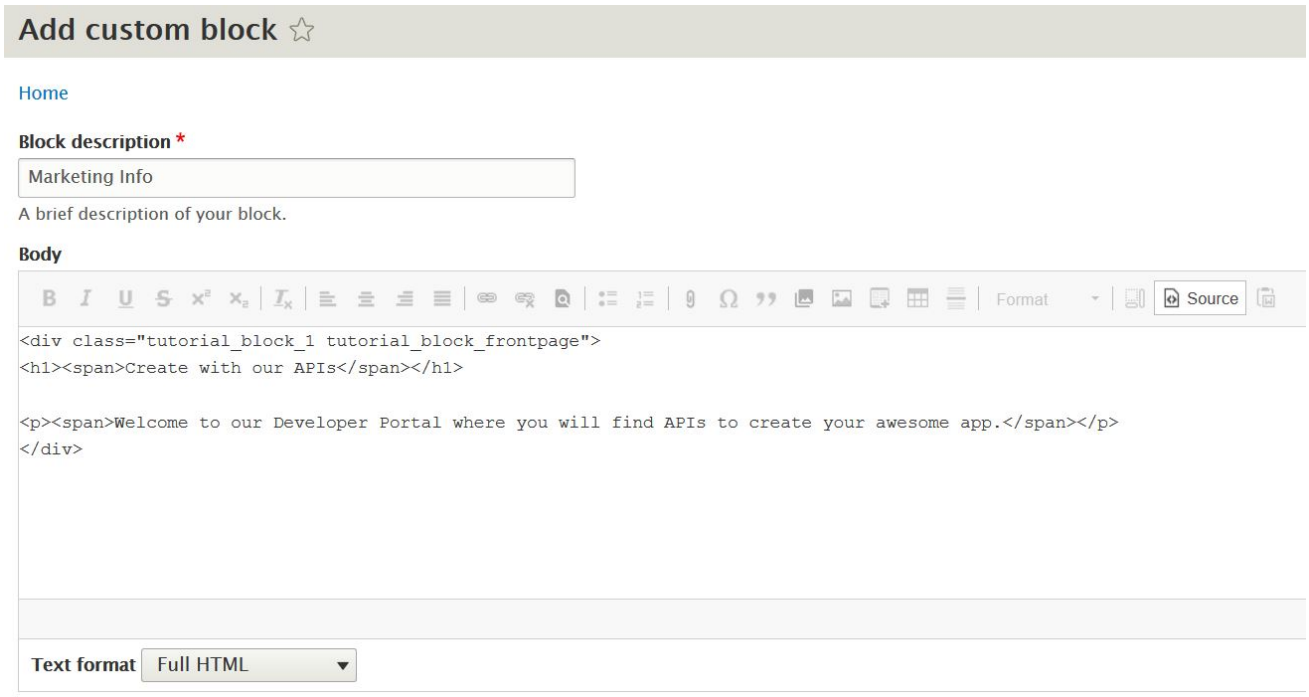
The front page displays like this before the block is added.



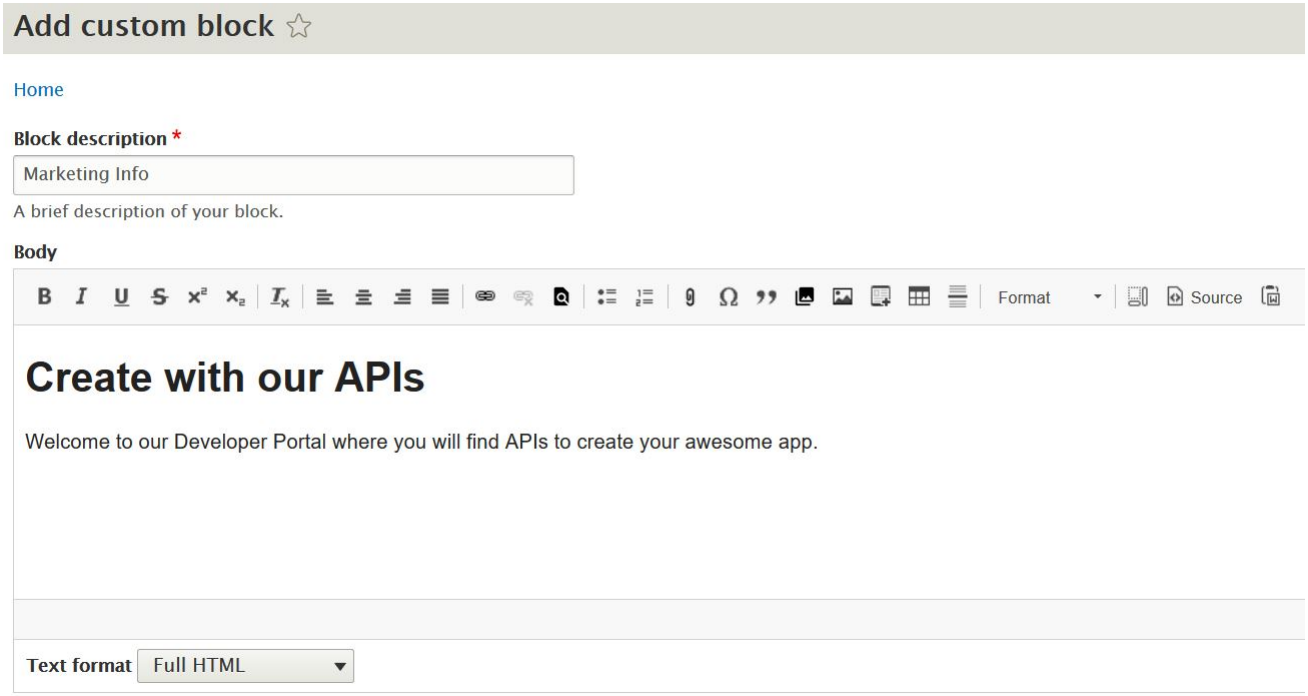
Creating a custom block

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Block layout > Add custom block.
4. Enter `Marketing Info` in the **`Block description`**.
5. In the body area, click **`Source`**, then enter this HTML:

```
<div class="tutorial_block_1 tutorial_block_frontpage">
<h1><span>Create with our APIs</span></h1>
<p><span>Welcome to our Developer Portal where you will find APIs to create your awesome app.</span></p>
</div>
```



6. To see a preview of the block click **`Source`** again.



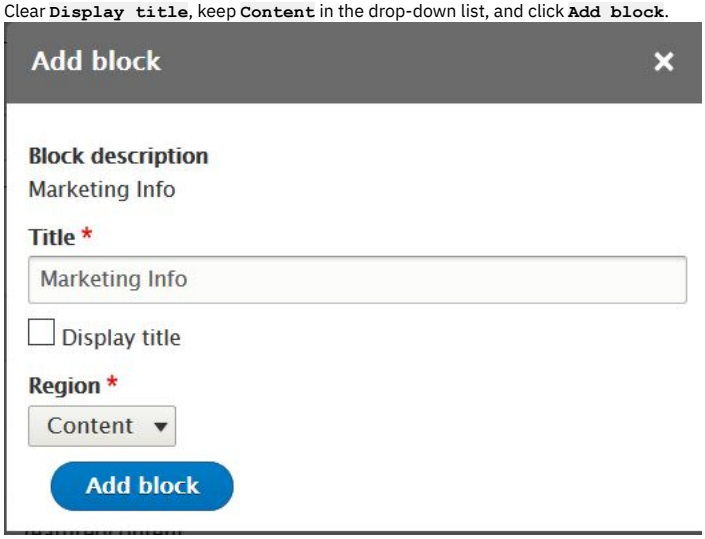
Note: This block uses almost no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see Tutorial: Creating a custom theme for the Developer Portal.

7. Click **`Save`**.

You have now created the custom block.

Adding the custom block to the front page

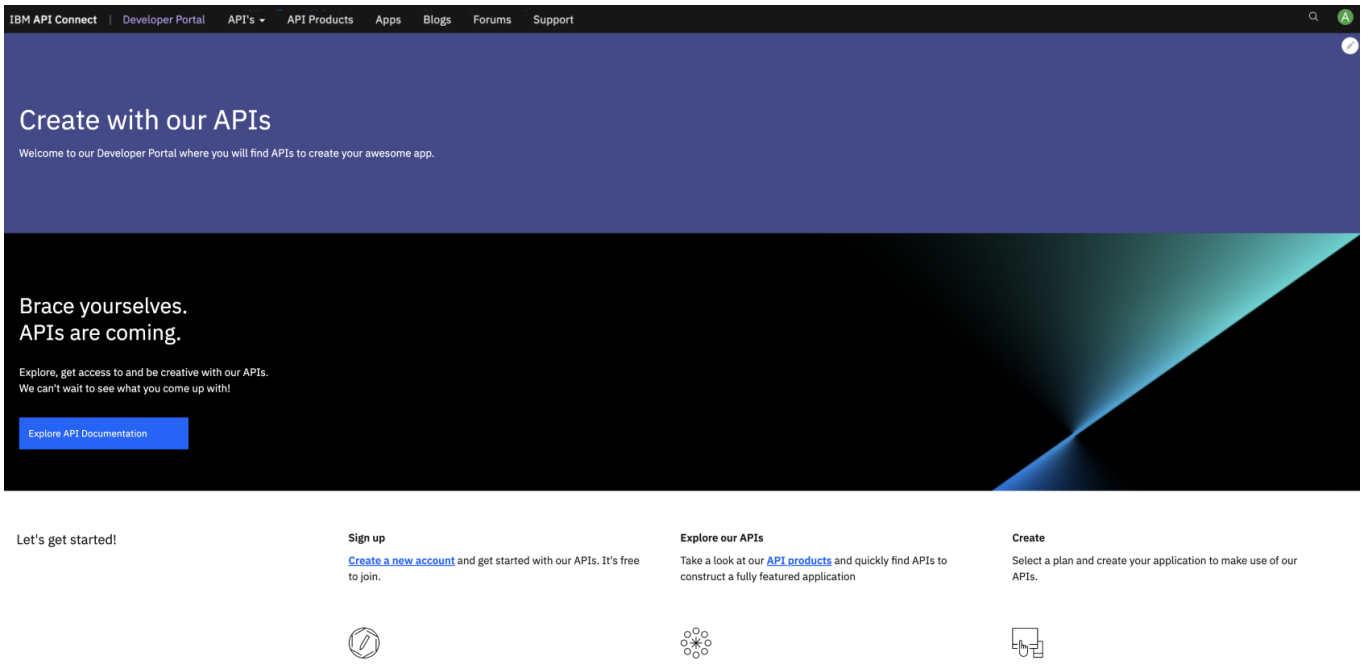
1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Pages.
3. Click **`Edit`** on the Welcome page label.
4. Click **`Panels`** in the **`Variants`** section.
5. Click **`Content`** on the expanded menu.
6. Click **`+ Add new block`**.
7. Click **`Marketing Info`** in the Custom list.
8. Clear **`Display title`**, keep **`Content`** in the drop-down list, and click **`Add block`**.



9. Drag **`Marketing Info`** to the order you want, in this example before the **`Welcome banner`** content.
10. Click **`Update and save`**.

What you did in this tutorial

The front page now has the new block with a marketing message.



What to do next

You can add further branding by adding content to your custom blocks such as images, or try using one of the provided custom blocks such as the Featured Content Block. Instead of using inline styling like that provided in the html, adopt best practice by adding class selectors and developing a custom theme. By default this block is English, you might want to consider adding translated versions.

Tutorial: Adding a custom block to a page other than the front page

You can customize the appearance of a page on your Developer Portal by adding a custom block. Blocks are boxes of content that are rendered into an area or region of a web page, such as the Products page or Apps page. Blocks can be displayed in regions, such as footer or sidebar, on your page.

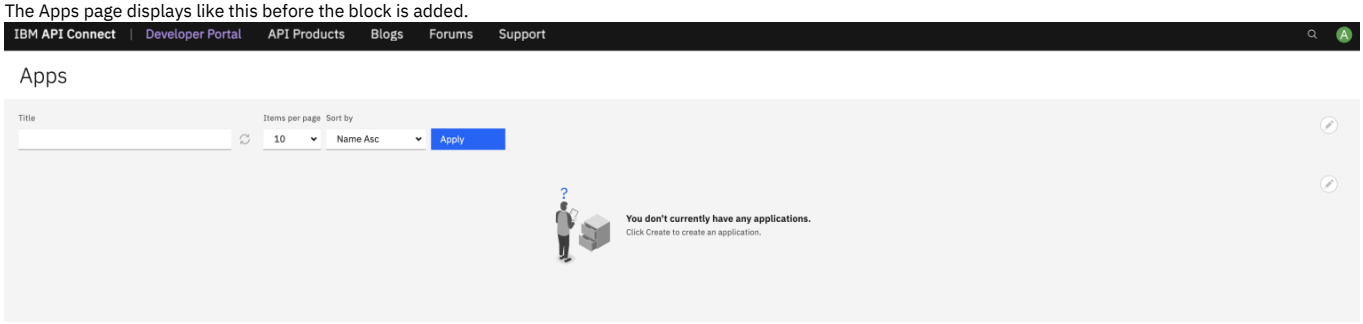
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so. The tutorial Adding a custom block to the front page explains how to add a custom block to the front page.

About this tutorial

In this tutorial, you create a custom block that contains extra guidance on application creation and registration, and add that custom block to the Apps page of your Developer Portal.

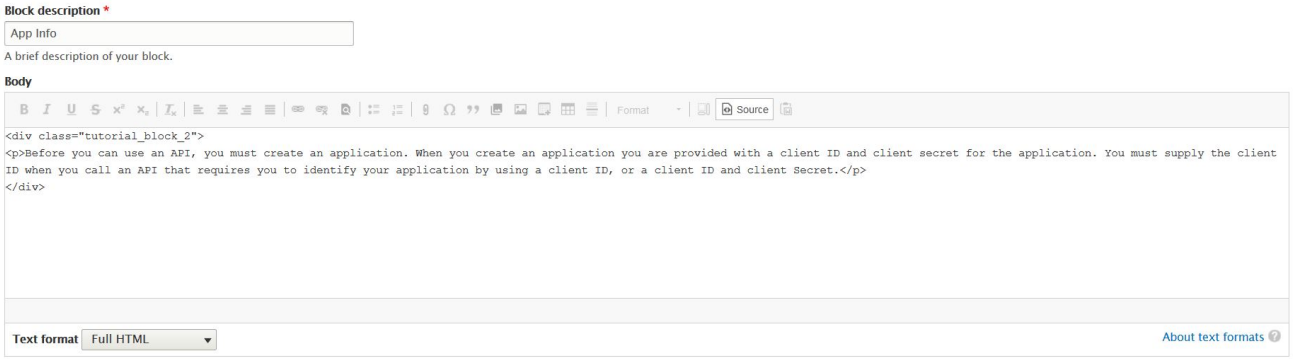
The Apps page displays like this before the block is added.



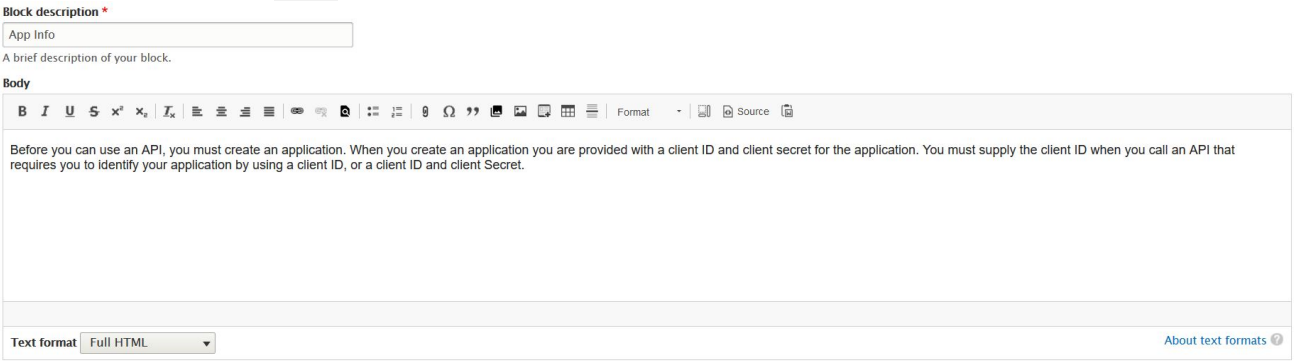
This tutorial takes you through creating a custom block, and configuring it to show on the Apps page.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Structure > Block layout > Add custom block.
4. Enter `App Info` in the **`Block description`**.
5. In the body area, click **`Source`**, then enter this HTML:

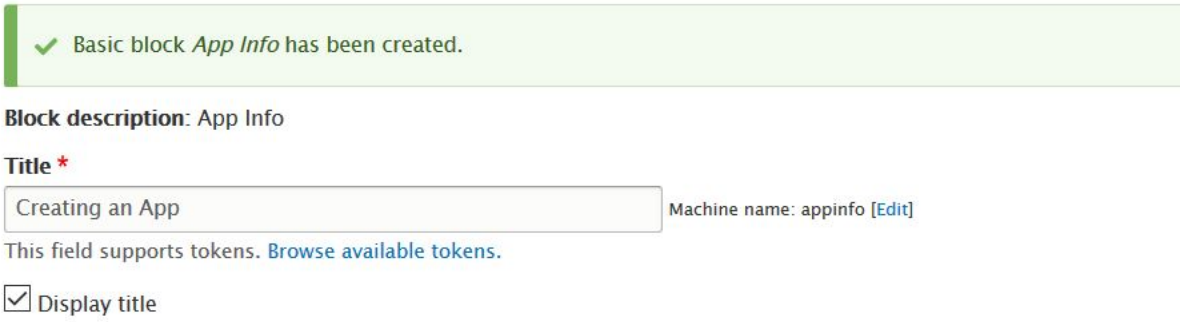
```
<div class="tutorial_block_2">
<p>Before you can use an API, you must create an application. When you create an application you are provided with a client ID and client secret for the application. You must supply the client ID when you call an API that requires you to identify your application by using a client ID, or a client ID and client Secret.</p>
</div>
```



6. To see a preview of the block click **`Source`** again.



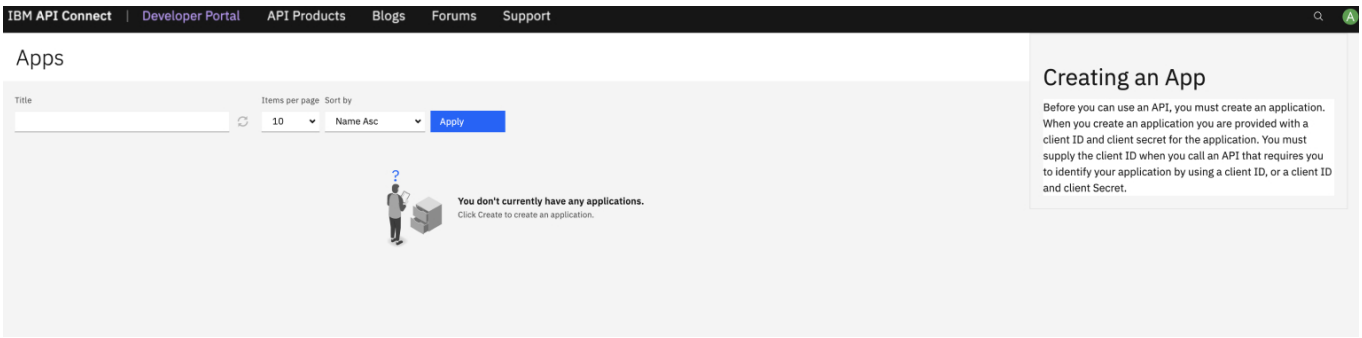
   Note: This block uses almost no inline-styling, instead it allows your theme to decide how it is rendered. The advantage of this set up is that you can easily change the rendering without modifying existing content in the database. Any styling that is required can be added to a custom theme. For more information, see Tutorial: Creating a custom theme for the Developer Portal.
7. Click **`Save`**.
8. Enter `Creating an App` in the **`Title`** box, keep the **`Display title`** selected.



9. Click **`Pages`** in the **`Visibility`** list.
10. Enter `/application` in the **`Pages`** text area, keep the **`Show for the listed pages`** selected.
11. Select **`Secondary`** from the **`Region`** drop-down menu.
12. Click **`Save block`**.
13. Click **`Save blocks`**.

What you did in this tutorial

The Apps page now has an additional block with a guidance message.



What to do next

Instead of using inline styling like that provided in the html, adopt best practice by adding class selectors and developing a custom theme. By default this block is English, so you might want to consider adding translated versions.

Tutorial: Configuring the RobotsTxt file

You can control the access of a visiting Web robot. You can configure the **`robots.txt`** file that exists on your web server, usually at the root level, to control access. Web robots are programs that crawl through the web to obtain web content for all the sites that are visited, and provide indexing for better performance of search engines. You can also specify separate rules for different robots.

Why would I want to edit Drupal's pre-existing robots.txt file?

Malicious robots might choose not to honor the **`robots.txt`** file, and by editing this file you are broadcasting which sites you do not want others to see. Therefore, you should not use this file to hide sensitive data. Instead, you might want to edit your **`robots.txt`** file to:

- Prevent duplicate information from being identified on your site
- Prevent internal pages from appearing in search engines
- Prevent private pages from appearing in search engines
- Prevent particular images, files, and so on, from bring crawled
- Specify a **`crawl-delay`** attribute to prevent robots from overloading your server at load time
- Exclude a particular robot

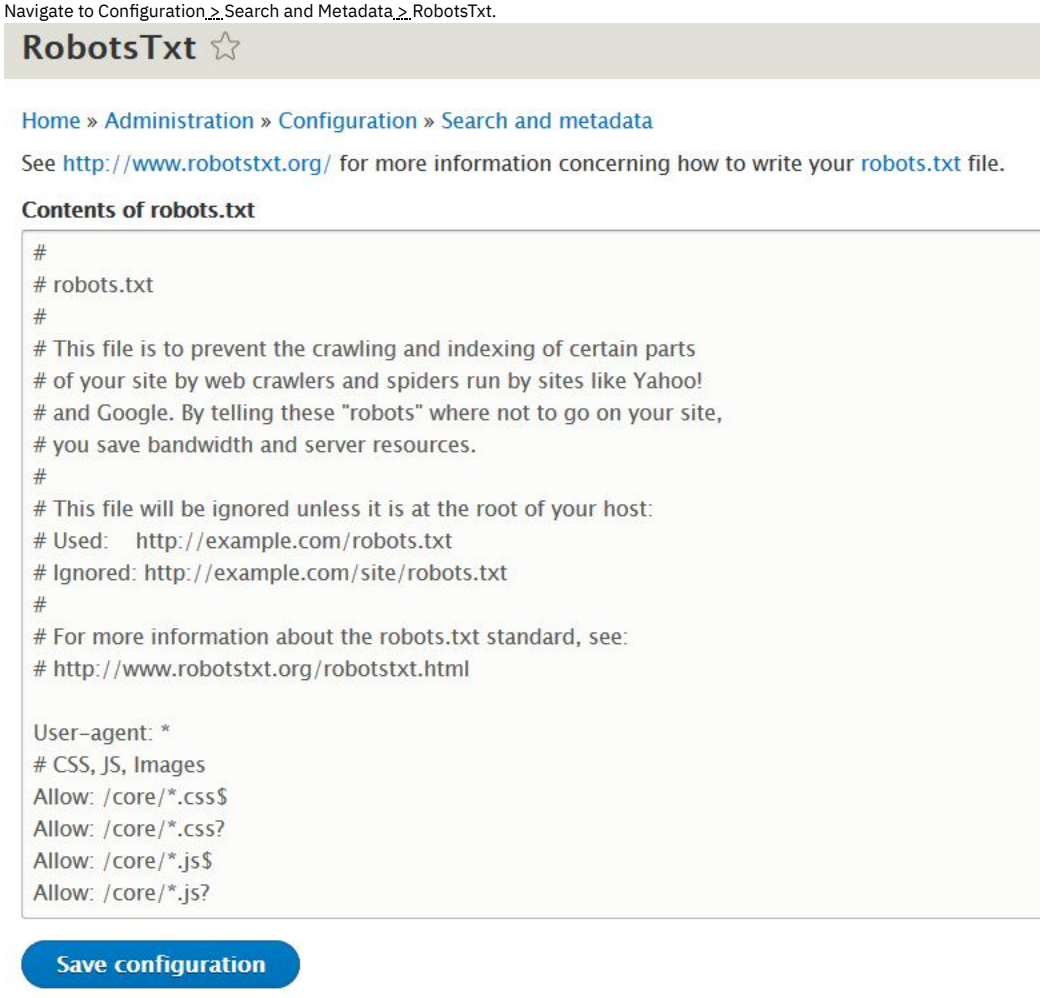
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

About this tutorial

You will edit the pre-existing **`robots.txt`** file and exclude access to a visiting robot called BadBot.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > Search and Metadata > RobotsTxt.



4. Enter the policy to exclude access to a robot called BadBot.

```
User-agent: BadBot
Disallow: /
```

5. Click **Save Configuration** to save your changes.

What you did in this tutorial

You have now successfully customized the **robots.txt** file. Robots now use this updated file to decide where they can crawl on your site. The BadBot robot is excluded access.

You can check whether your **robots.txt** file is successfully changed by navigating to your site and appending /robots.txt. You should see the content you entered into that file.

```
#
# robots.txt
#
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used:    http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/robotstxt.html

User-agent: BadBot
Disallow: /

User-agent: *
# CSS, JS, Images
Allow: /core/*.css$
Allow: /core/*.css?
Allow: /core/*.js$
Allow: /core/*.js?
```

For more information on how to edit your **robots.txt** file, see https://www.robotstxt.org/.

What to do next

You can edit the **robots.txt** at any time by navigating back to the page within the configuration settings. You might choose to duplicate this file across all of your sites, or choose different policies for different sites.

Tutorial: Configuring the SecurityTxt file

You can use the **security.txt** file to provide your users with a standard way to report security vulnerabilities with your site.

The **security.txt** file is a file format that is designed to help your users disclose any security vulnerability. For more information, see securitytxt.org or A File Format to Aid in Security Vulnerability Disclosure.

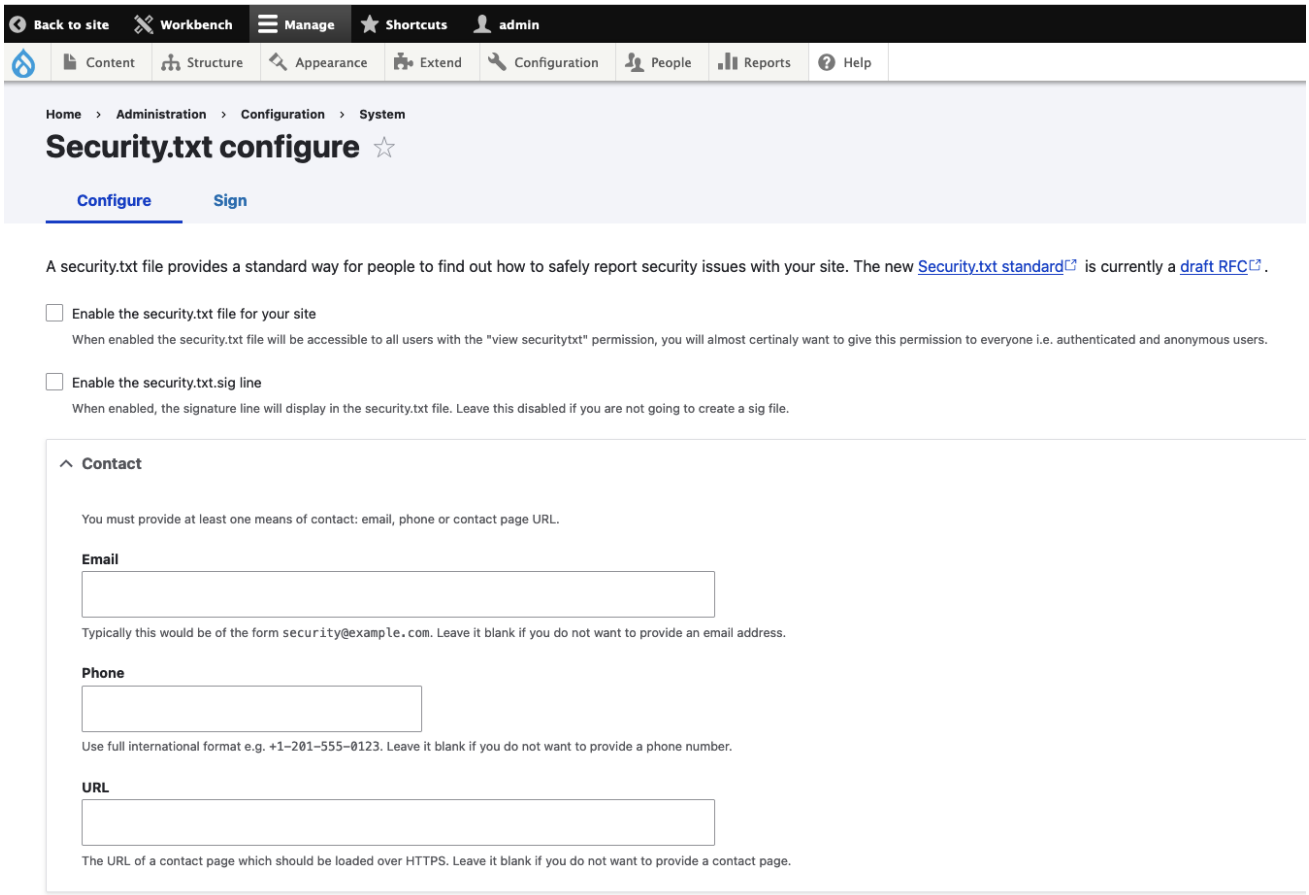
Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

About this tutorial

You edit the **security.txt** file and provide information such as a URL of a contact page, for your customers to use if they have a security vulnerability.

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Navigate to Configuration > System > Security.txt.



4. Select the checkbox for **`Enable the security text file for your site`**.
5. Complete the form for your requirements.
6. Click Save Configuration to save your changes.

What you did in this tutorial

You successfully customized the **`security.txt`** file.

What to do next

You can edit the **`security.txt`** at any time by navigating back to the page within the configuration settings.

Tutorials for using custom modules in the Developer Portal

You can configure custom modules to complete specific actions in the Developer Portal.

Note: You must have administrator access to complete the tutorials.
You can configure rules for the following events that occur in the Developer Portal:

- Change the profile page to display **`firstname last name`**, instead of **`username`**.
- Add validation to a field on the sign up form.

Tutorial: Changing the profile page to display firstname lastname, instead of username

You might want to change the profile page of your users on your Developer Portal to display their **`firstname`** and **`lastname`** rather than their **`username`**.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial.

About this tutorial

In this tutorial, you change the profile page of your users to display their **firstname** and **lastname** rather than their **username**. The change to the profile page is done by using the custom module **change account display** that is already available in IBM APIC Portal - addons. However, for more information, see Custom module development: creating a module skeleton.

This tutorial takes you through the following steps:

1. Cloning and packaging the **change account display** module.
2. Installing and enabling the **change account display** module.

Cloning and packaging the change account display module

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

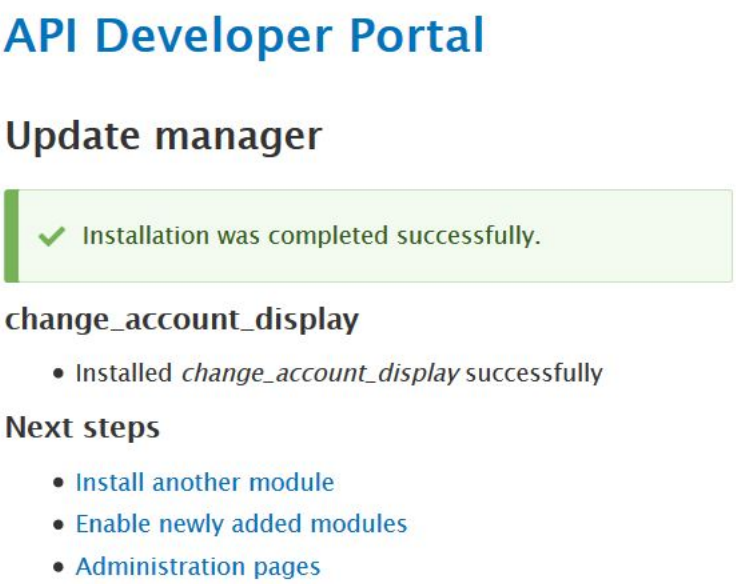
   ```
   git clone https://github.com/ibm-apiconnect/devportal-addons
   ```

2. Package up the custom module.

   ```
   cd devportal-addons/apic_v10/modules
   tar -czf change_account_display.tar.gz change_account_display/
   ```

Installing and enabling the change account display module

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend > Install new module.
4. Click **Browse** from the **Upload a module or theme archive to install** section.
5. Navigate to and select the packaged custom module that you created earlier. For example, **change_account_display.tgz**, then click **Open**.



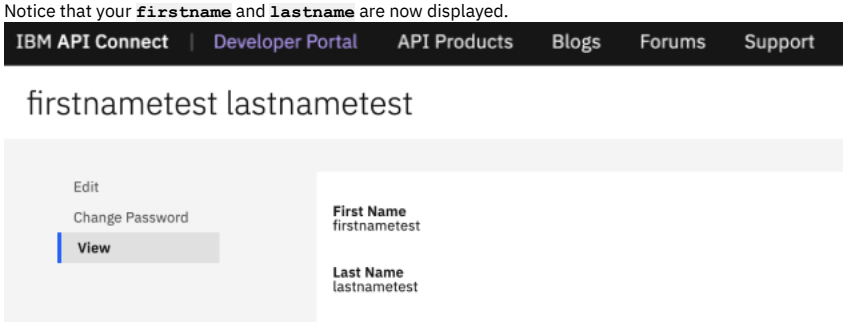
6. Click **Install**.
7. Click the **Enable newly added modules** link.
8. Enter **Change Account Display** in the filter box.
9. Check the check box next to **Change Account Display**.
10. Click **Enable**.

The module is now installed and enabled.

Test the profile page change

1. Sign out as the Admin user.
2. Sign in as a user of your Developer Portal.
3. Click **My account** from the drop down next to your **username**.

4. Notice that your **firstname** and **lastname** are now displayed.



What you did in this tutorial

You used a custom module to change the profile page display for the users of your Developer Portal.

Tutorial: Adding validation to a field on the sign-up form

You might want to add extra validation to a field to help your Developer Portal users. For example, you might want to validate a department name, employee ID, or similar data when a user registers for an account on the portal.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. You must have completed the tutorial about Adding a field to the sign up form, where you added a Department code field to the Sign-up form.

About this tutorial

In this tutorial, you add some validation to the Department code field on the Sign-up form. This is a required field for all portal users and is a string of the format **DEPnnn**, where **DEP** is the string prefix and **nnn** is a three-digit number, for example, **DEP123**. The validation is done by using a custom module that is already available in IBM APIC Portal - addons. However, for more information, see Custom module development: creating a module skeleton.

This tutorial takes you through the following steps:

1. Cloning and packaging the validation module.
2. Installing and enabling the validation module.
3. Testing the validation in the sign up form.

Look at the files we will use in this link IBM APIC Portal - addons.

- user_field_example.info.yml - this is the metadata for the module, defining things like the name and version, and containing other important information that is required by Drupal to import and enable the module.

```
name: 'IBM APIC Portal - user field example'
type: module
description: 'IBM API Connect Developer Portal tutorial - Example of validating a custom user field'
package: 'Custom'
core_version_requirement: ^8 || ^9 || ^10
version: 1.0.0
project: 'user_field_example'
dependencies:
  - ibm_apim
  - auth_apic
```

- user_field_example.module - this contains a validation function and adds this to the Sign-up form by using the **hook_form_alter** function.

```
<?php
use Drupal\Core\Form\FormStateInterface;

/**
 * Implementation of hook_form_alter() to alter the Sign up form
 */
function user_field_example_form_alter(&$form, FormStateInterface $form_state, $form_id) {
  if ($form_id === 'user_register_form' ) {
      $form['#validate'][] = 'user_field_example_validate_department_code';
  }
}

/**
 * Validate the Department code field on the Sign up form.
 *
 * Valid entry = DEPnnn
 *   where n = single figure digit.
 */
function user_field_example_validate_department_code($form, &$form_state) {
  $dept_code = $form_state->getValue('field_department_code')[0]['value'];
  $valid = preg_match('/^DEP\d{3}$/', $dept_code);
  if (!$valid) {
    $form_state->setErrorByName('field_department_code', t('Invalid department code.'));
```

```
  }
}
```

Cloning and packaging the validation module

Validation is added to the **Department code** field on the Sign-up form by using a custom module.

1. Clone the custom module. Here is an example command if you are using Mac and Linux:

   ```
   git clone https://github.com/ibm-apiconnect/devportal-addons
   ```

2. Package up the custom module.

   ```
   cd devportal-addons/apic_v10/modules
   tar -czf user_field_example.tgz user_field_example/
   ```

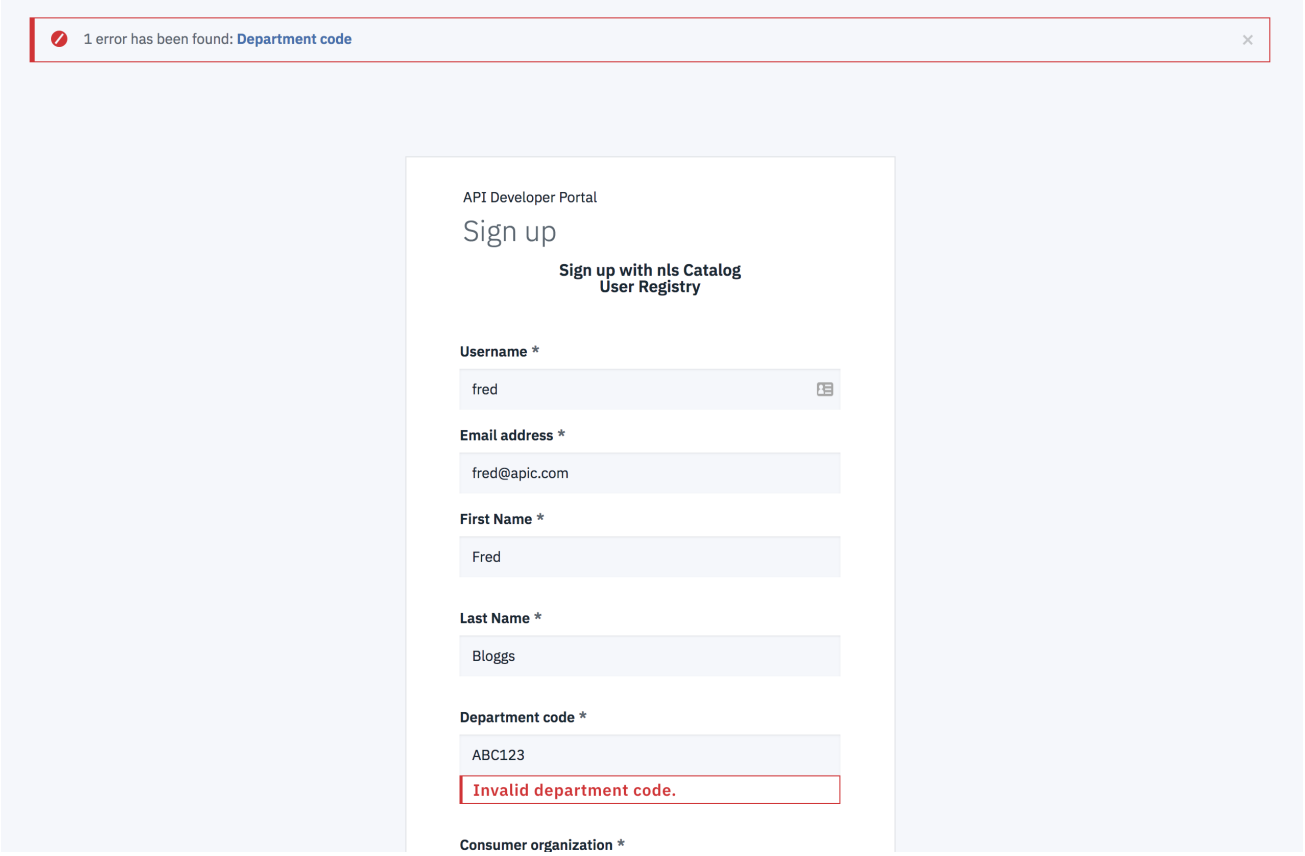
Installing and enabling the validation module

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Extend > Install new module.
4. Click **Browse file** from the **Upload a module or theme archive** section.
5. Browse to and select **user_field_example.tgz**, the packaged custom module that you created earlier. Click **Open**.
6. Click **Continue**.
7. Click the **Enable newly added modules** link.
8. Enter **IBM APIC Portal - user field example** in the filter box.
9. Check the checkbox next to **IBM APIC Portal - user field example**.
10. Click **Enable**.

The module is now installed and enabled.

Testing the validation in the sign-up form

1. Sign out as the Admin user.
2. Click **Create account**.
3. Complete all fields, set **Department code** to **ABC123**.
4. Click **Sign up**.
5. Observe the failure message and highlighting on the **Department code** field.



6. Change the **Department code** field to **DEP245**, reenter the password and click **Sign up** to see that a valid code is accepted.

What you did in this tutorial

You added some validation to the Department code field on the Sign-up form, then tested the form to ensure that the **`Department code`** field accepts only codes in the format of **`DEPnnn`**.

In reality, you would need to add this configuration in such a way that it is available for all sites. To achieve this availability, you would include the configuration in your custom module. For more information, see Drupal custom modules.

Tutorial: Using a custom weighting sort order on the product list page

You can use a custom weighting to sort your products on the API products list page of your Developer Portal.

Before you begin

You must have a Developer Portal enabled, and you must have administrator access to complete this tutorial. The tutorial Creating the Portal explains how to enable the portal if you have not already done so.

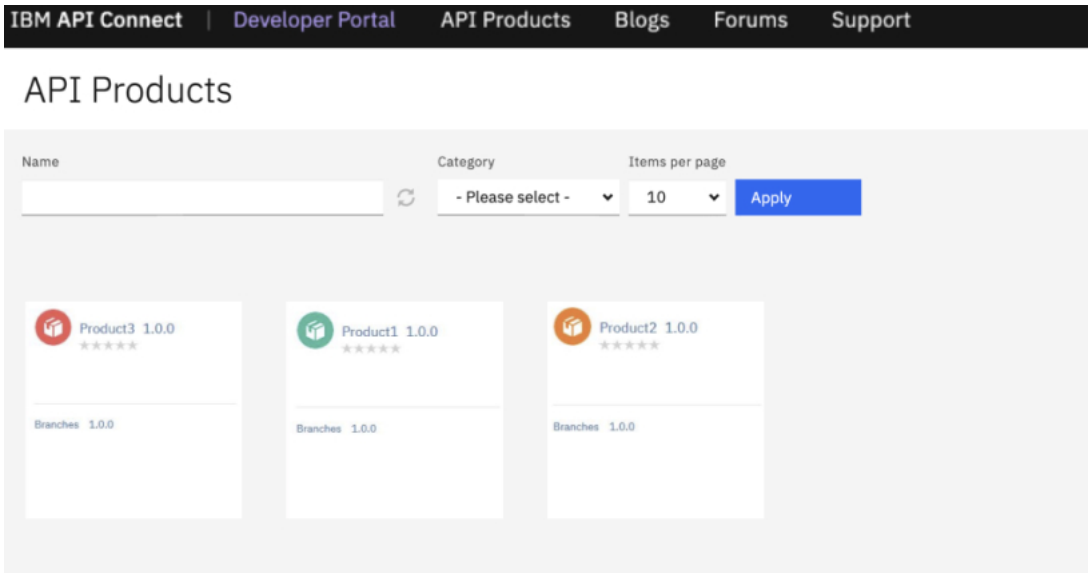
About this tutorial

In this task, you set up a weighting sort criteria and use it to list your products, on the API Products page, in a particular order. You set up the weightings so the order reads, Product3, Product1, and Product2.

This tutorial takes you through the following steps:

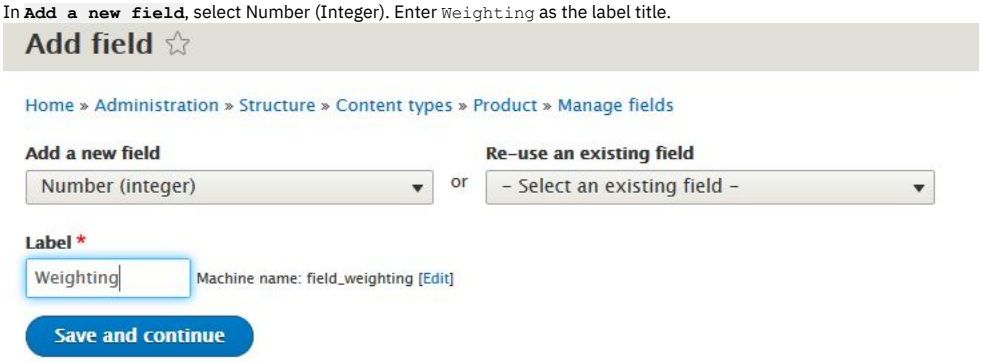
1. Add a custom integer field to the API content type.
2. Modify the Products view to change the sort order.
3. Add a weighting to each Product.

Note: To complete this tutorial, you will need to have API products.
For example, the following API Products page displays Product1, Product2, and Product3 in this order before the weightings are added.



Add a custom integer field to the API content type

1. Log in to your Developer Portal as an administrator.
2. If the administrator dashboard isn't displayed, click Manage to display it.
3. Click Structure > Content types > Product > Manage fields.
4. Click + Add field.
5. In **`Add a new field`**, select Number (Integer). Enter `Weighting` as the label title.

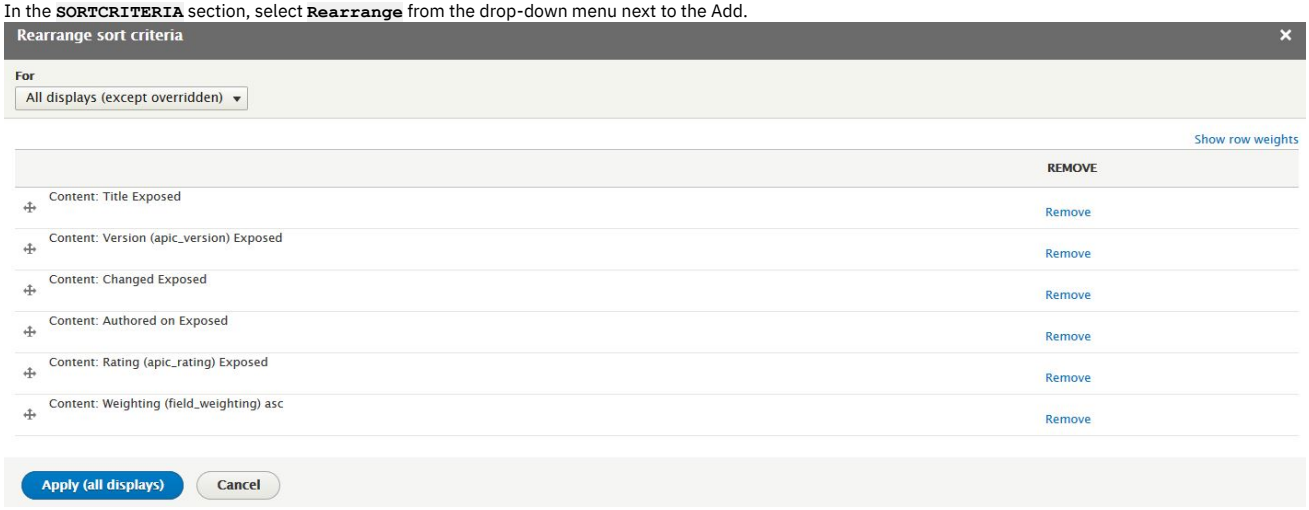


6. Click Save and continue.
7. Leave the **Allowed number of values** set to 1 and click Save field settings.
8. Enter 0 as the default value, click Save settings.
9. Click Save.

The **Weighting** field now shows in the **Manage fields** list.

Modify the Products view to change the sort order

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Navigate to Structure > Views, scroll down to the **Products** view and click Edit.
3. In the **SORT CRITERIA** section click Add.
4. Enter weight in the search box. Then, select **Weighting (field_weighting)**, and click Apply (all displays).
5. Leave the **Order** as **Sort ascending**, click Apply (all displays).
6. In the **SORTCRITERIA** section, select **Rearrange** from the drop-down menu next to the Add.

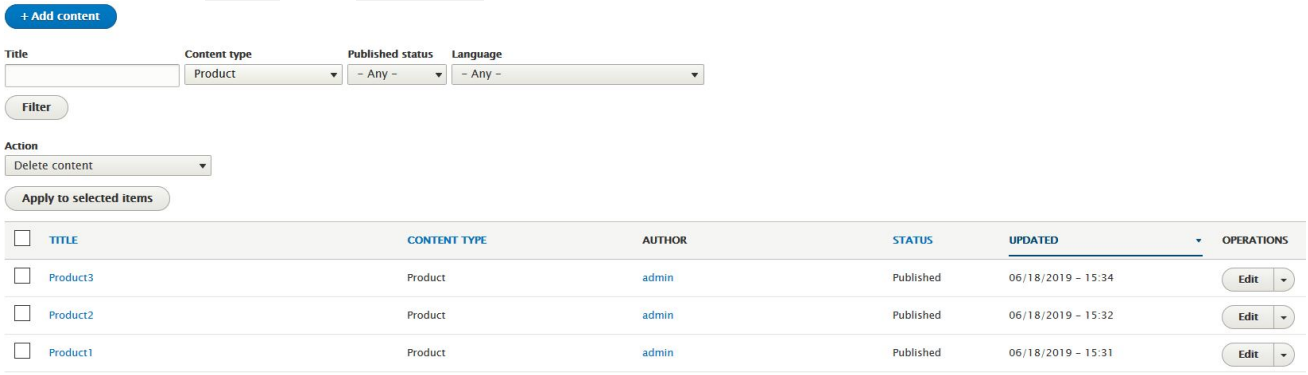


7. For all rows except **Weighting (field_weighting)**, click **Remove**.
8. Click Apply (all displays).

You modified the **Products** view to sort based on **Weighting (field_weighting)**.

Add a weighting to each Product

1. If the administrator dashboard isn't displayed, click Manage to display it.
2. Click Content, and select **Product** from the **Content type**, click Filter. A list of available Products on your site is displayed.

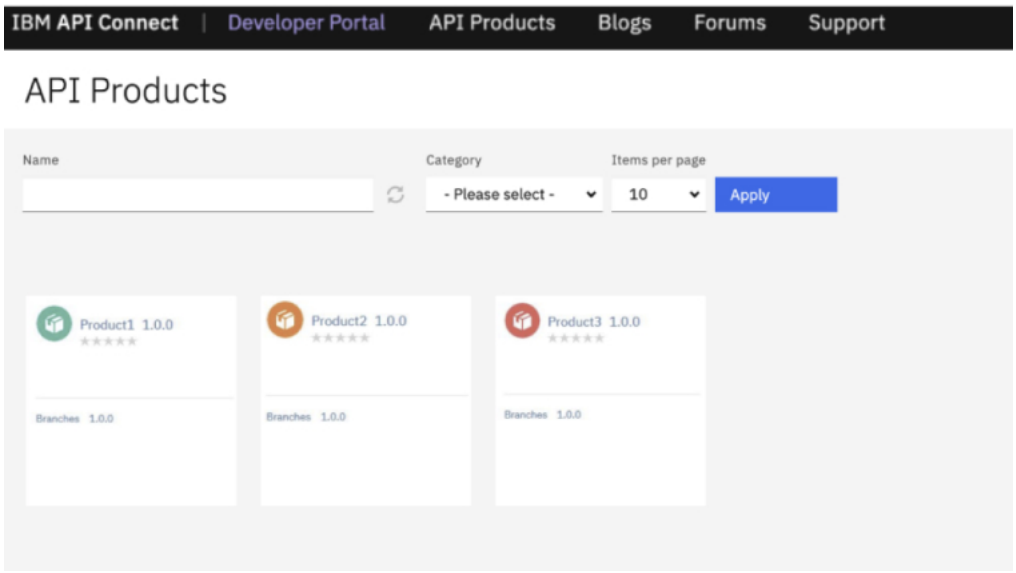


3. For Product1 click edit, in the **Weighting field**, select 2 as the value, click Save.
4. For Product2 click edit, in the **Weighting field**, select 3 as the value, click Save.
5. For Product3 click edit, in the **Weighting field**, select 1 as the value, click Save.

The API Products page displays Product3, Product1, and Product2 in this order after the weightings are added.

What you did in this tutorial

You changed the order that the API Products appear on the products page.



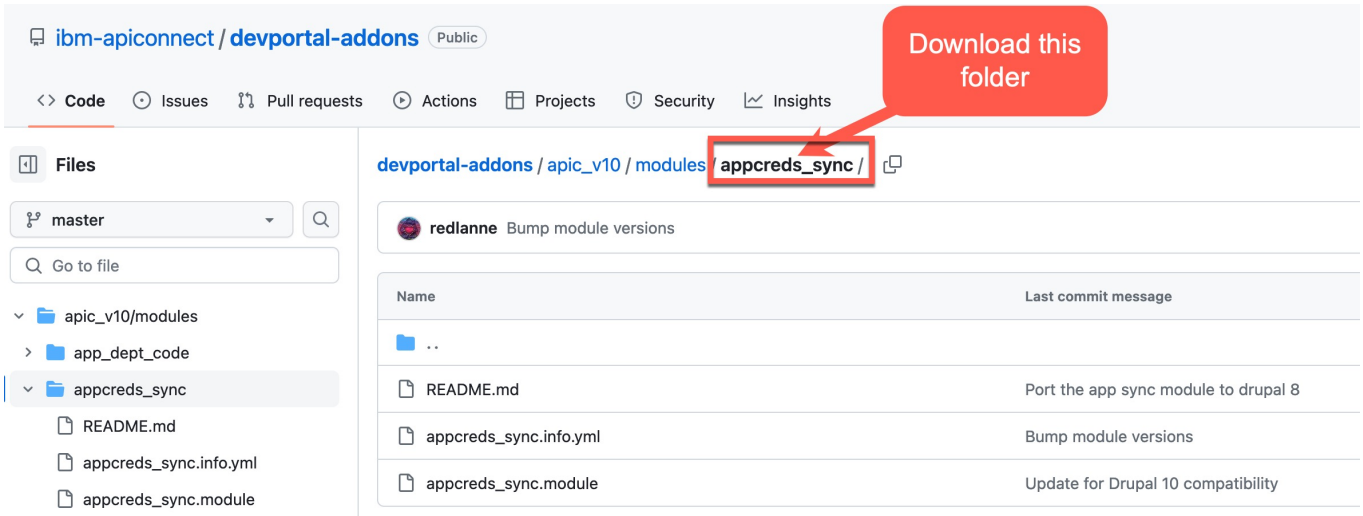
Tutorial: Synchronizing application credentials from the Developer Portal with an external server

You can synchronize application credentials from the Developer Portal with an external server.

Before you begin

You must have administrator access to complete this task.

Download the Application credential synchronization folder.



Perform the following steps:

1. Clone the custom module. Here is an example if you are using Mac and Linux:

   ```
   git clone https://github.com/ibm-apiconnect/devportal-addons
   ```

2. Navigate to the `appcreds_sync` folder:

   ```
   cd devportal-addons/apic_v10/modules/appcreds_sync
   ```

About this task

In this tutorial, you configure a custom module and install it in the Developer Portal to synchronize application credentials between the Developer Portal and an external credentials server.

The **`appcreds_sync`** folder that you downloaded is a custom module that contains multiple methods that can be configured with your REST endpoints, and any further specifications that you want to define, to synchronize your Developer Portal application credentials with an external credential server.

For more information, see Custom module development: background and prerequisites.

Procedure

To configure the custom module to synchronize application credentials:

1. In the appcreds_sync.module file, replace `example.com` with your REST endpoint for the following method types:
   - `http://example.com/app/create`
   - `http://example.com/app/delete`
   - `http://example.com/app/creds/create`
   - `http://example.com/app/creds/update`
   - `http://example.com/app/creds/delete`
   - `http://example.com/app/clientid/reset`
   - `http://example.com/app/clientsecret/reset`
2. Optional: Replace `username` and `password` with your username and password in the `_appcreds_sync_json_http_request` function:

   ```
   ('username' . ':' . 'password')
   ```

   Note: You can also configure the SSL settings if you need to.
3. Save the file, and package up the custom module:

   ```
   cd..
   tar -czf appcreds_sync.tar.gz appcreds_sync/
   ```

To install the configured custom module in the Developer Portal:

4. Log in to your Developer Portal as an administrator.
5. If the administrator dashboard does not display, click Manage.
6. Click Extend > Install new module.
7. In the Add from a URL field, enter a path to the module . Alternatively, upload the module in the Upload a module or theme archive section.
8. Click Continue.

To enable the module:

9. If the administrator dashboard does not display, click Manage.
10. Click Extend and search for the custom module just installed, for example, Synchronise Application Credentials.
11. Select the check box and click Enable.

Results

You have configured a custom module that synchronizes application credentials from the Developer Portal, installed that module in the Developer Portal, and enabled it. No further action or configuration is required.

Analytics: understand your API usage

Monitor your API usage with API Connect analytics.

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

- **Key concepts of API Connect analytics**
  An overview of how API Connect analytics works.
- **Preinstallation planning**
  Review the analytics subsystem requirements and decide how you want to configure analytics for your environment.
- **Managing your analytics subsystem**
  Understand the architecture of the analytics subsystem and how to configure it.
- **Installing the analytics subsystem**
  Links to the installation steps for each platform.
- **Backup, restore, and disaster recovery**
  How to back up your analytics subsystem and restore after a disaster.
- **API event record field reference**
  Reference table and examples of the data that is stored in an API event record.
- **Advanced configuration options**
  Configure advanced features and settings such as offload, retention period, and dedicated storage.
- **Customize your analytics data**
  How to customize the data stored in your API event records.
- **Access your analytics data**
  How to view, analyze, and export your analytics data.
- **Troubleshooting analytics**
  How to diagnose and resolve common issues with the analytics subsystem.

Key concepts of API Connect analytics

An overview of how API Connect analytics works.

How the gateway and analytics services work together

When a user calls one of your published APIs, the gateway that processes the API call creates an API event record that contains details of the API call such as the application ID, current time, result code, etc. For a full list of all the data stored in an API event record, see API event record field reference. The gateway buffers API event records for each API call until a threshold of either number of events, or elapsed time is reached. When this threshold is reached, the API events are sent in bulk to the analytics subsystem.
Tip: If the analytics dashboards in your UIs are not showing the API events that you expect, the most common cause is a communication failure between your gateway and analytics subsystems. Check both the gateway and analytics subsystem logs.

You can deploy multiple gateway and analytics services in all API Connect deployment types except for OpenShift top-level CR. After you register a gateway service in the Cloud Manager UI, you associate an analytics service with that gateway service. All API event data that the gateway generates is sent to its associated analytics service. Multiple gateways can be associated with the same analytics service. You cannot associate multiple analytics services with the same gateway.

Each analytics service in your API Connect deployment stores API event data only from the gateways it is associated with. When you view analytics data, you must select which analytics service you want to view the data from. API Connect provides no aggregated view of all analytics data from all analytics services. If you have multiple analytics services and want to combine the data from all of them, you can export or offload the data and manually create your own reports.

If you use the top-level CR on OpenShift or Cloud Pak for Integration, then your API Connect deployment consists of one gateway service and one analytics service. All API calls in your API Connect deployment are handled by the one gateway service, and all API event records that are generated, are stored by the one analytics service.

Customizing the analytics data that is stored for your APIs

Note: What API consumer analytics are displayed in the Developer Portal depends on the visibility configuration settings. For more information, see Configuring analytics in the developer portal.
When you configure an API in the API Manager UI, you choose what data is stored in the API event records for calls to that API with the **`activity log`** property. The **`activity log`** property has separate settings for successful API calls, and for failed API calls. The activity log options are:

- none: API events are not logged.
- activity: Logs the API resource URI. **`activity`** is the default log setting for successful API calls.
- header: Logs the API resource URI and HTTP headers.
- payload: Logs API resource URI, HTTP headers, and request and response payloads. **`payload`** is the default log setting for failed API calls, since the response payload might have useful information on the cause of the failure.

The more detailed the **`activity log`** setting, the larger the API event records, and the greater the persistent storage requirements for the analytics subsystem. For more information about configuring activity logging, see Activity log policy.

If you want to make specific customizations to your API event data, two methods are available:

- Customize API event records as they arrive from the gateway, by using ingestion filters. Ingestion filters can be applied conditionally based on the data in the API event.
- Customize API event records as they are created on the gateway. Use a gateway script or the **`set-variable`** operation in the definitions of each API that you want to customize. See Customizing data with a gateway script or set-variable operation.

Retention and rollover of analytics data

API event data takes up persistent storage space on your analytics subsystem. To ensure that you do not run out of persistent storage space, API event data is summarized, and the oldest API event records are deleted. Analytics API event records are retained for 30 days on a one replica deployment, and 90 days on a three replica deployment. Summary analytics data that is created from the API event records is retained for 1 year. You can change how long the analytics data is stored with the Retention and rollover settings.

Accessing your analytics data

You can view your analytics data in the Cloud Manager, API Manager, and Developer Portal UIs.

- The cloud admin user can see total API call volumes and call volumes per gateway in the Cloud Manager UI. Provider organization owners can enable the cloud admin user to also see analytics data and individual API event data for their provider organization.
- In the API Manager UI, the provider organization owner can see analytics data for APIs that are published in their catalogs.
- In the Developer Portal UI, the API consumer can see analytics data for APIs published in their consumer organization.
  Note: If two gateway services are configured for a catalog that has a portal site, those two gateway services must be associated with the same analytics service. If they are associated with different analytics services, then no analytics data is displayed in the portal UI.
  This limitation means that analytics data is not available in the developer portal sites of a two data center disaster recovery deployment, where two analytics subsystems are deployed.

The maximum number of API events that can be accessed from the UIs is 10,000. The maximum number of event records that can be displayed per page in the UIs is 200.

Your analytics data can also be accessed by using the toolkit CLI or REST API.

You can manage user access to analytics data with User roles and permissions. For more information about user-based analytics data visibility, see Catalogs, spaces, and analytics

Exporting your analytics data

You can export analytics data from your UIs, however, UI export is not a practical method if you want to export more than 200 API event records. Use the toolkit CLI or REST API to export more than 200 API event records at a time. If you think that you might want to regularly export all your analytics data, then consider Offloading your analytics data to a third-party system.

Offloading your analytics data to a third-party system

The API event data that is received from your gateways can be directed to a third-party system, instead of or in addition to being stored by the analytics subsystem.

For more information about offloading, see Offload to third-party systems.

Preinstallation planning

Review the analytics subsystem requirements and decide how you want to configure analytics for your environment.

- **Estimating storage requirements**
  Estimate the persistent storage requirements for your analytics subsystem.
- **Kubernetes and OpenShift preinstallation planning**
  Review deployment profiles, storage, and analytics settings that must be decided before installation.
- **VMware preinstallation planning**
  Review deployment profiles and other analytics settings that must be decided before installation.

Estimating storage requirements

Estimate the persistent storage requirements for your analytics subsystem.

This information applies only to data that is stored in your API Connect analytics subsystem, and not to data that is offloaded to a third-party system. If you plan to disable local storage, you can skip this topic. The formula and guidelines are based on how the API Connect analytics subsystem stores data, and cannot be directly applied to any other storage system.

Warning: Increasing the analytics storage space after installation is not a simple operation, and it can require reinstallation of the analytics subsystem. If you do not have the information that is needed to make an estimate, then set your storage to at least 500Gi and monitor your usage closely. For an installation where you expect to store little or no analytics data, you can set the storage to 50Gi.
After installation, if you find you do not have enough space, then reduce the data retention period to keep less analytics data and stay within your persistent storage limit.

To understand the concepts that are required for estimating your storage requirements, first review the following pages:

- OpenSearch nodes, indices, shards, and replicas.
- Activity log policy.
- Retention and rollover.

Use the following guidelines to provide an estimate of the amount of persistent storage you need for your API Connect analytics subsystem.

Factors that determine persistent storage requirement

Number of copies of the analytics data
: The number of copies of the analytics data you store depends on your deployment profile:

  - Three replica profiles store 3 copies of your analytics data (one OpenSearch shard replica per analytics replica).
  - One replica profiles store 1 copy of your analytics data.

Number of days that API data is retained
: The default number of days your API event data is retained depends on your deployment profile:

  - Three replica profiles retain API event data for up to 90 days.
  - One replica profiles retain API event data for up to 30 days.

  API summary data is retained for 366 days by default on all deployment profiles.

  If you customized your retention and rollover settings, then use the updated retention periods in your calculations.

API event logging policy and customization
: The logging policy that you set for each API determines the size of the event records for that API. Typical sizes of API event records for each activity log policy are:

  - Activity logging: 1.5 Kb. This is the default log policy for successful API calls.
  - Header logging: 4 Kb.
  - Payload logging: 4 Kb plus size of expected request and response payloads. This is the default log policy for failed API calls.

  You can also Customize your analytics data to add, redact, or remove fields. If you add fields to your API events, calculate the average size of the new fields and add that number to your event size. If you remove fields, subtract this size only if you are also removing headers or payloads.

Percentage of each type of data
: Estimate the percentage of each type of log policy (activity, header, payload) for your published APIs. Use these percentages and your estimates of API event size to calculate you average API event size.
  Remember: API logging policy can be set differently for success and failure responses of the same API. The default is activity logging for successful API calls, and payload logging for failed calls. Consider the percentages of successful and failed calls that you expect in your calculations.

API calls per day
: Estimate the total number of API calls that you expect per day.

Calculating your persistent storage requirements

Calculate the expected size of your API event data with the following formula:

```
Storage required for API events = (average bytes per API event) * (API events per day) * (API event retention period) * (copies of data)
```

Calculate the expected size of your API summary data with the following formula:

```
Storage required for API summary data = (expected size of all API events with header logging) * (API events per day) * (summary data retention period) * (copies of data) * 0.1
```

Note: Ignore the percentages of different logging types you calculated for your APIs and assume that all your APIs are set to header logging. API summary data uses approximately 10% of the space of header-logged API event data per day. Remember that header logging includes everything that is logged by activity logging plus the API call headers.
Your analytics data must not use more than 70% of your persistent storage space. Add some buffer space in case your analytics data is larger than you expect. It is recommended that the persistent storage space you provide for your analytics subsystem is double the space that you calculated your data requires.

Example calculation, based on the following information:

```
Copies of data = 3
API event data retention = 90 days
API summary data retention = 366 days
Activity log bytes per call = 1500 bytes
Header log bytes per call = 15000 bytes
Payload log bytes per call = 30500 bytes
% of Activity log = 70%
% of Header log = 20%
% of Payload log = 10%
Calls per day = 100000
```

1. Calculate the average API event size:

   ```
   ( (70*1500) + (20*15000) + (10*30500) ) / 100 = 7100 bytes
   ```

2. Calculate the space needed for your API event data:

   ```
   7100 * 100000 * 90 * 3 = 178.5 Gi (191.7 Gb)
   ```

3. Calculate the space needed for your summary data:

   ```
   15000 * 100000 * 366 * 3 * 0.1 = 153 Gi (165 Gb)
   ```

4. Add API event data and summary data totals:

   ```
   153 Gi + 178.5 Gi = 331.5 Gi
   ```

5. Double the storage calculated in step 4 to get the storage size to allocate to your analytics subsystem:

   ```
   331.5 Gi * 2 = 663 Gi
   ```

Kubernetes and OpenShift preinstallation planning

Review deployment profiles, storage, and analytics settings that must be decided before installation.

Deployment profile

Decide which deployment **`profile`** you want to use, the available profiles are:

- For Kubernetes and OpenShift (individual subsystem) installations: Analytics deployment profiles.
- For Cloud Pak for Integration and OpenShift top-level CR installations: API Connect deployment profiles.

For a scalable analytics deployment, choose a three replica profile. If you do not expect a high API transaction rate and plan to keep a limited amount of analytics data, then a one replica profile is sufficient.

Storage class

Before you install analytics, you must choose a storage class. **`Ceph`**, **`Block`**, and **`Local Volume`** are all supported; however, **`Local Volume`** is the most suitable for analytics. The analytics subsystem provides internal data replication and HA support, so the additional HA capabilities that **`Ceph`** or **`Block`** storage might give you are not needed. In addition, analytics requires a high throughput of disk I/O operations to successfully handle the analytics data load from the gateway. **`Local Volume`** is the best-performing storage class for disk I/O.

Note: GlusterFS and NFS storage are not supported for analytics. If you use either of these storage classes, you might encounter severe performance degradation, and possibly loss of data.

Storage type

Note: Storage type selection is available on three replica deployments only.
To make an informed choice on which storage type is best for you, first read and understand the OpenSearch concepts that are described in OpenSearch nodes, indices, shards, and replicas.
API Connect provides two OpenSearch storage types:

- **Shared storage**
  A single pod provides data storage and OpenSearch cluster management. Shared storage is the default option, and the only option on a one replica deployment.

- **Dedicated storage**
  OpenSearch data storage and cluster management functions are split, and run in separate pods on each worker node:
  - The **`storage-os-master`** pod runs an OpenSearch "manager eligible" node that manages the OpenSearch cluster, but does not store any API event data.
  - The **`storage`** pod runs an OpenSearch "data" node that stores API event data.

Dedicated storage facilitates horizontal scaling to support greater analytics data storage and throughput. Add worker nodes and increase the number of **`storage`** pod replicas to scale horizontally (you do not need to increase the number of **`storage-os-master`** replicas). Dedicated storage also provides greater stability and allows some OpenSearch configuration changes to be made without downtime.
You can change storage type after installation, see Dedicated storage and scaling up.

Disabling internal storage

The analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution if you are using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

When you disable internal storage for analytics data, the following microservices are disabled:

- `osinit`
- `storage`

Important:
If you disable local storage, the analytics views in your API Connect UIs remain accessible, but are empty.

Configure local storage disablement before installation. It is not possible to disable or re-enable internal storage after installation.

You can enable offload of analytics data to third-party systems after installation.

For the steps to disable internal storage, see Disable local storage.

Persistent storage space

Based on your expected API transaction rate, you can estimate how much storage space your analytics subsystem requires. See Estimating storage requirements.

Inter-subsystem communication security

By default the network communication from the management and gateway subsystems, to the analytics subsystem uses Kubernetes ingresses or OpenShift routes, and is secured with mTLS. You can configure alternative network communication options if your environment requires. For more information, see Network requirements for inter-subsystem communication.

Kubernetes operating map counts

When the analytics service is configured to store data, it uses OpenSearch, which requires map counts higher than the operating system defaults. The minimum recommended value is `262144`. Unless you plan to disable local storage, increase the default map count on every Kubernetes worker node:

1. To change the map counts on the live system, run the following command on every Kubernetes node:

   ```
   sudo sysctl -w vm.max_map_count=262144
   ```

2. To persist this change when node restarts occur, add the following setting to the /etc/sysctl.conf file:

   ```
   vm.max_map_count = 262144
   ```

For more information, see Important settings in the OpenSearch documentation.

PV requirements

The number of PVs needed depends on if you disable local storage.

Table 1. Analytics PV requirements

Deployment profile	Local storage enabled	Local storage disabled (ingestion only)
Single node (n1)	2	1
Three node (n3)	6	3

To understand how much storage capacity your analytics subsystem needs, see Estimating storage requirements.

Firewall requirements

The analytics firewall requirements are documented here: Firewall requirements

VMware preinstallation planning

Review deployment profiles and other analytics settings that must be decided before installation.

Deployment profile

The first decision that you must make is which `deployment-profile` you want to use. The available deployment profiles for analytics are: `n1xc2.m16`, `n1xc4.m32`, `n1xc6.m48`, `n3xc4.m16`, `n3xc4.m32`, `n3xc6.m48`, `n3xc8.m64`. For more information about deployment profiles, see Deployment profiles.

For a typical production deployment, choose a three replica profile. A one replica profile is only sufficient if you expect a low API transaction rate and do not plan to retain much analytics data.

Storage space

Based on your expected API transaction rate, you can estimate how much storage space your analytics subsystem requires. See Estimating storage requirements.

Firewall requirements

The analytics firewall requirements are documented here: Firewall requirments.

Inter-subsystem communication security

If your network infrastructure requires that load-balancers implement TLS termination, then mTLS between API Connect subsystems can be disabled and JSON Web Token (JWT) security can be used instead.
Note: Although mTLS is disabled, the network communication is still secured with standard TLS, which does not require passthrough to be enabled on load-balancers.
For more information about configuring JWT for network security, see: Enable JWT security.

Disabling internal storage

If you are offloading your analytics data to a third-party system, you can choose to not store your API event data in your API Connect analytics subsystem by disabling local analytics storage. With local storage disabled, the CPU, memory, and storage requirements of your analytics subsystem are greatly reduced, but you lose the ability to view your analytics data in the API Connect UIs.
Important: It is not possible to enable or disable local storage after installation.

Managing your analytics subsystem

Understand the architecture of the analytics subsystem and how to configure it.

- **API Connect on VMware**
  An overview of the analytics VM, and how to configure analytics features.
- **API Connect on Kubernetes, OpenShift, and Cloud Pak for Integration**
  Understand the Kubernetes artifacts that make up the analytics subsystem, and how to configure your AnalyticsCluster CR.
- **OpenSearch nodes, indices, shards, and replicas**
  Understand the main concepts of the OpenSearch database: nodes, indices, shards, and replicas.

API Connect on VMware

An overview of the analytics VM, and how to configure analytics features.

Analytics data storage

The disk size for analytics data is specified at installation when you deploy your analytics OVA file, see Deploying the analytics subsystem OVA file.

You can increase the space available for your analytics data by adding a disk, see Adding disk space to a VMware appliance.
Warning: Do not attempt to increase the size of the existing data disk. Changing the disk size corrupts your existing analytics data.

Managing the analytics subsystems VMs

Manage your analytics VMs with the **apicup** command from your project directory. Some analytics settings are specified directly as arguments to the **apicup** command, for example:

```
apicup subsys set <analytics subsystem> <key> <value>
```

Other analytics settings are configured in a YAML file that is called `analytics-extra-values.yaml`, see Analytics extra-values file.

Some maintenance operations require SSH access to your analytics VMs. To access you analytics VMs with SSH, follow these steps:

1. Use an SSH client to login as the `apicadm` user:

   ```
   ssh apicadm@<analytics VM FQDN>
   ```

2. If you need root user access, `sudo` to root:

   ```
   sudo -i
   ```

CAUTION:
Do not do any operations inside the analytics VM that are not described in the API Connect documentation or otherwise advised by IBM.

Analytics extra-values file

Create a file called analytics-extra-values.yaml to configure the following analytics features:

- Logstash persistent queue
- Offload to third-party systems
- Customizing data with ingestion filters
- Disable local storage
  Note: If you want to disable local storage, you must configure it **before** installation. You cannot change this setting after installation.

To create and apply the analytics-extra-values.yaml file, follow these steps:

1. Create a file called analytics-extra-values.yaml in your project directory.
2. The analytics-extra-values.yaml contains a single object called **spec**. Set the first line of the file to:

   ```
   spec:
   ```

   Define your analytics features inside the **spec** object. Use 2 spaces for indentation. For example, to disable local storage, set your analytics-extra-values.yaml as follows:

   ```
   spec:
     external:
       offload: # enable offload to third-party system
         enabled: true
         output: |
           http {
             url => "example.com"
             http_method => "post"
             codec => "json"
             content_type => "application/json"
             headers => {
               "x-acme-index" => "8"
               "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
             }
             id => "offload_http"
     storage:
       enabled: false # disable local storage
   ```

   Note: You can define one or more analytics features in the **spec** object. Do not leave the analytics-extra-values.yaml with an empty **spec** section.
3. Use the **apicup** command to assign your analytics-extra-values.yaml file to the **extra-values-file** property:

   ```
   apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
   ```

   Note: If **extra-values-file** is already set to analytics-extra-values.yaml, you do not need to rerun this command every time you edit analytics-extra-values.yaml.
4. Apply the analytics-extra-values.yaml file to your analytics subsystem with **apicup install**:

   ```
   apicup subsys install <analytics subsystem>
   ```

   Warning: Do not apply an empty analytics-extra-values.yaml file, or one that contains just the **spec:** line.

Firewall requirements

Your analytics VMs must be reachable on the ports that are described in API Connect firewall rerunquirements.

Analytics subsystem TLS certificates

No certificate-related sysadmin operations are normally required on your analytics VMs. The analytics subsystem has no user-facing certificates. For more information about certificates on VMware deployments, see API Connect TLS certificates best practices.

Extra values file reference

List of all the properties you can set in the **analytics-extra-values.yaml**.

Table 1. Fields that you can set in the **analytics-extra-values** file

Field	Type	Description	Comments
external: offload: filter	String	Filter configuration for external offload.	See Offload to third-party systems.
external: offload: enabled	Boolean	Enables analytics offload feature.	See Offload to third-party systems.
external: offload: output	String	Output configuration for external offload.	See Offload to third-party systems.
external: offload: passwordSecretName	String	The name of the secret that has the password to access the certificate keystore.	See Offload to third-party systems.
external: offload: secretName	String	The name of the secret that contains the certificates used for connecting to the offload target.	See Offload to third-party systems.
ingestion: filter	String	Filter to modify API event data that is stored internally (not offloaded).	See Customizing data with ingestion filters.
ingestion: queue: type	Object	Persistent queue settings. Default is **persisted**.	Set to **memory** if you want to disable the persistent queue. See Logstash persistent queue.
ingestion: queue: volumeClaimTemplate: volumeSize	String	Storage size for your persistent queue. Default is 50Gi.	Do not change this value. See Logstash persistent queue.

Field	Type	Description	Comments
`ingestion: queue: volumeClaimTemplate: storageClassName`	String	Storage class name for your persistent storage. **Must** be set to local-storage.	See Logstash persistent queue.
`ingestion: pipelineWorkers`	4 <= integer <= 128	Number of pipeline workers.	See Performance tuning with ingestion pipeline properties.
`ingestion: pipelineBatchSize`	10 <= integer <= 10000	Batch size of a pipeline worker.	See Performance tuning with ingestion pipeline properties.
`ingestion: pipelingBatchDelay`	10 <= integer <= 5000	Batch delay of a pipeline worker.	See Performance tuning with ingestion pipeline properties.
`ingestion: configReloadAutomatic`	Boolean	Whether to automatically reload the pipeline on configuration changes.	See Performance tuning with ingestion pipeline properties.
`ingestion: pipelineUnsafeShutdown`	Boolean	Whether to allow unsafe shutdown of the pipeline.	See Performance tuning with ingestion pipeline properties.
`storage: enabled`	Boolean	Set false to disable internal storage.	See Disable local storage.

API Connect on Kubernetes, OpenShift, and Cloud Pak for Integration

Understand the Kubernetes artifacts that make up the analytics subsystem, and how to configure your AnalyticsCluster CR.

The AnalyticsCluster Custom Resource (CR)

Your analytics subsystem is defined by a Custom Resource (CR) called `AnalyticsCluster`. Throughout this documentation, the `AnalyticsCluster` CR is also referred to as the "analytics CR".

Certain analytics features are configured in the `AnalyticsCluster` CR. How you edit the CR depends on your platform:

- **Kubernetes and OpenShift individual subsystem CRs:** Edit the `AnalyticsCluster` directly.
- **Cloud Pak for Integration and OpenShift top-level CR:** Edit the analytics section of the top-level CR. When saved, the changes to the top-level CR are applied to `AnalyticsCluster` CR. If you change the `AnalyticsCluster` CR directly, changes can be overwritten by the top-level CR.

Note: For convenience, the alias `a7s` can be used instead of `AnalyticsCluster` when you edit the CR. For example:

```
kubectl edit a7s
```

instead of:

```
kubectl edit AnalyticsCluster
```

The properties that you can configure in the `AnalyticsCluster` CR are listed in Analytics CR properties reference.

Ingresses

The analytics subsystem has one ingress, called the "ingestion endpoint". The ingestion endpoint serves requests coming from both the management subsystem and the gateway. The management subsystem communicates with the ingestion endpoint to query API event data stored by the analytics subsystem, and also to manage the analytics subsystem. The gateway sends API event data to the ingestion endpoint.

The ingestion endpoint is listed as `ai-endpoint` or `<instance name>-ai-endpoint` in the output of `kubectl get ingress`.

Note: In OpenShift, "routes" are used instead of ingresses. The functions of the ingestion endpoint are the same for both the Kubernetes `ai-endpoint` ingress and the OpenShift `ai-endpoint` route.

Pods

The following table shows the Kubernetes pods that make up an analytics subsystem:

Table 1. Analytics pods

Name	Description
`osinit`	OpenSearch initialization job. This pod is in `completed` state after installation.
`oscron`	Manages the jobs that transform API event data into summary data. Runs every 15 minutes.
`director`	Manages the analytics subsystem. The director pod serves all requests that come from the management subsystem, such as: initial subsystem registration, queries for analytics data, management of saved queries, and changes to retention settings.
`ingestion`	Logstash-based data pipeline that receives the API event data from the gateway, applies any ingestion filters that the user configures, and forwards the events to the storage pods and/or to third-party offload targets.
`mtls-gw`	Secures all communication that arrives on the analytics ingestion endpoint, forwarding communication from the management subsystem to the `director` pod, and communication from the gateway to the `ingestion` pod.

Name	Description
`storage`	The `storage` pods contain the OpenSearch database that stores all the analytics data. By default the storage pods also run the OpenSearch cluster management tasks. If dedicated storage is enabled the OpenSearch cluster management tasks are done by the `storage-os-master` pods, and the `storage` pods contain just the analytics data.
`storage-os-master`	The `storage-os-master` pod exists only on deployments where dedicated storage is enabled. The `storage-os-master` pods manage the OpenSearch cluster, while the `storage` pods are dedicated to the storage of the analytics data.

PersistentVolumeClaims (PVCs)

The following table describes the PVCs that are created for the analytics subsystem:

Table 2. Analytics PVCs

Name	Description
`data-analytics-storage-<integer>`	The PVC that stores your analytics data. A `data-analytics-storage-<integer>` PVC is created for each analytics `storage` pod. The `<integer>` is the replica number, starting from 0.
`data-analytics-storage-os-master-<integer>`	If dedicated storage is enabled, then the `storage-os-master` PVCs are created. Each `storage-os-master` PVC corresponds to a `storage-os-master` pod. The `<integer>` corresponds to the pod replica number, starting from 0. The PVC stores data that is related to the management of your OpenSearch cluster.
`data-analytics-ingestion-<integer>`	If the persistent queue feature is enabled, a `data-analytics-ingestion-<integer>` PVC is created for each `ingestion` pod. The `data-analytics-ingestion-<integer>` PVC stores API event data temporarily when the `ingestion` pod is unable to forward the data to the `storage` pods. The `<integer>` corresponds to the pod replica number, starting from 0.

Certificates

The following table describes the certificates that are used by the analytics subsystem:

Table 3. Analytics certificates

Certificate name	Issuer	Description
`analytics-ai-endpoint or a7s-ai-endpoint`	`ingress-issuer`	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the `analytics-ai-endpoint` certificate. If this certificate is updated, restart the `mtls-gw` pod for the update to take effect.
`analytics-ca or a7s-ca`	`selfsigning-issuer`	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.
`analytics-client or a7s-client`	`analytics-ca`	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`analytics-server or a7s-server`	`analytics-ca`	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`<instance name>-storage`
		
		If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

For more information about certificates in API Connect, see API Connect certificates.

Analytics CR properties reference

List of all the properties that can be configured in the analytics CR.

Note: If you have a VMware deployment, do not edit the analytics CR directly. Use the **apicup** command and analytics-extra-values.yaml file. See Analytics extra-values file.
The following properties must be defined in your analytics CR:

- `ingestion.endpoint`
- `ingestion.clientSubjectDN`

All other properties are optional.
Note: If you are using Cloud Pak for Integration or the top-level CR on OpenShift, then configure your analytics properties in the `analytics` section of the top-level CR. Do not edit the analytics CR directly. If the properties you want to set are missing from the top-level CR, then copy them from the analytics CR to the top-level CR.

Table 1. Analytics subsystem CR properties.

Property	Type	Default value	Description	Comments
`allowUpgrade`	Boolean	false	Legacy property for upgrade to v10.0 5 from previous releases.	The `allowUpgrade` property is not used in v10.0.6 and can be deleted from the analytics CR.
`databaseBackup:` `  host`	String		The URL to your S3 object store.	See Backup, restore, and disaster recovery.
`databaseBackup:` `  path`	String		The path to your S3 object store at the URL specified by `databaseBackup.host`.	See Backup, restore, and disaster recovery.

Property	Type	Default value	Description	Comments
`databaseBackup: schedule`	String		Specifies the frequency of automatic backups. Use any valid `cron` string.	See Backup, restore, and disaster recovery.
`databaseBackup: enableCompression`	Boolean	`true`	Specifies whether to compress the backups.	See Backup, restore, and disaster recovery.
`databaseBackup: credentials`	String		The name of your S3 object store authentication secret.	See Backup, restore, and disaster recovery.
`databaseBackup: backupCerts`	String		The name of the secret that contains your S3 server certificate.	See Backup, restore, and disaster recovery.
`databaseBackup: chunkSize`	String	`1gb`	Specifies the size of large file chunks. Format is lower case and no spaces, for example: `1gb`, `50mb`, `800kb`.	See Backup, restore, and disaster recovery.
`databaseBackup: enableServerSideEncryption`	Boolean	`true`	Specifies whether metadata files are stored in compressed format.	See Backup, restore, and disaster recovery.
`external: offload: filter`	String		Filter configuration for external offload.	See Offload to third-party systems.
`external: offload: enabled`	Boolean	`false`	Enables analytics offload feature.	See Offload to third-party systems.
`external: offload: output`	String		Output configuration for external offload.	See Offload to third-party systems.
`external: offload: passwordSecretName`	String		The name of the secret that has the password to access the certificate keystore.	See Offload to third-party systems.
`external: offload: secretName`	String		The name of the secret that contains the certificate that is used for connecting to the offload target.	See Offload to third-party systems.
`ingestion: endpoint`	Object		FQDN and TLS Secrets for client endpoint.	The external endpoint of the analytics subsystem. Both the gateway and management subsystems use the ingestion endpoint to communicate with your analytics subsystem. The gateway sends API event data to the ingestion endpoint.
`ingestion: clientSubjectDN`	String		Subject DN (distinguished name) for authenticating with ingestion endpoint (RFC 2253).	
`ingestion: filter`	String		Filter to modify API event data before it is sent for local storage (not offloaded).	See Customizing data with ingestion filters.
`ingestion: queue`	Object		Persistent queue settings.	See Logstash persistent queue.
`ingestion: queue: type`	String		Persistent queue settings. Default is `persisted`.	Set to `memory` if you want to disable the persistent queue. See Logstash persistent queue.
`ingestion: queue: volumeClaimTemplate: volumeSize`	String		Storage size for your persistent queue. Default is 50Gi.	Do not change this value. See Logstash persistent queue.
`ingestion: queue: volumeClaimTemplate: storageClassName`	String		Storage class name for your persistent storage.	See Logstash persistent queue.
`ingestion: pipelineWorkers`	4 <= integer <= 128	24	Number of pipeline workers.	See Performance tuning with ingestion pipeline properties.
`ingestion: pipelineBatchSize`	10 <= integer <= 10000	250	Batch size of a pipeline worker.	See Performance tuning with ingestion pipeline properties.

Property	Type	Default value	Description	Comments
`ingestion: pipelineBatchDelay`	10 <= integer <= 5000	50	Batch delay of a pipeline worker.	See Performance tuning with ingestion pipeline properties.
`ingestion: configReloadAutomatic`	Boolean	true	Whether to automatically reload the pipeline on configuration changes.	See Performance tuning with ingestion pipeline properties.
`ingestion: pipelineUnsafeShutdown`	Boolean	false	Whether to allow unsafe shutdown of the pipeline.	See Performance tuning with ingestion pipeline properties.
`longTermDataEnabled`	Boolean	true	Enables the collection of long-term summary data that is required for reports and leaderboards.	If you set `longTermDataEnabled` to false, no long-term summary data is generated, however the `oscron` job still runs. Long-term summary data is only generated when local storage is not disabled.
`standaloneCluster`	Boolean	false	Legacy property related to Zen endpoints.	The `standaloneCluster` property is not used in v10.0.6 and can be deleted from the analytics CR.
`storage: type`	String	`shared`	Type of storage deployment.	Shared or dedicated. See Storage type.
`storage: shared: volumeClaimTemplate: storageClassName`	String	cluster default	Storage class name.	Name of the storage class for your analytics data. See Storage class.
`storage: shared: volumeClaimTemplate: volumeSize`	String	`50Gi`	Volume size.	Size of the volume used for data storage. See Estimating storage requirements.

OpenSearch nodes, indices, shards, and replicas

Understand the main concepts of the OpenSearch database: nodes, indices, shards, and replicas.

OpenSearch do not use the term "database", but "search and analytics suite". For simplicity, in API Connect the terms "analytics database" or "OpenSearch database" are used instead of "search and analytics suite".

The term OpenSearch "cluster" is used when referring to the analytics subsystem of a multi-replica API Connect deployment.

Nodes

An OpenSearch cluster is made up of separate OpenSearch nodes. On Kubernetes, an analytics storage pod replica runs a single OpenSearch node. On VMware, a deployed analytics VM runs a single OpenSearch node.
You control the number of OpenSearch nodes that you have when you select your API Connect deployment profile: A one replica deployment has one OpenSearch node. A three replica deployment has three OpenSearch nodes.

If you installed a three replica deployment on Kubernetes or OpenShift, you can enable the dedicated storage feature to increase your number of OpenSearch nodes above three.

Indices

Your analytics API event data is stored in OpenSearch indices. The creation and management of your analytics indices is time-based. By default, your OpenSearch database creates a new index each day to be used for storing the API events that arrive that day; so on the day of installation your analytics subsystem has 1 index for API event data, and after 10 days your subsystem has 10 API event data indices. On a one replica API Connect deployment, API event indices are kept for 30 days. On a three replica installation, API event indices are kept for 90 days.
API Connect also creates indices for shared queries and for API summary data. By default API summary indices are kept for 1 year.

You can configure the retention and rollover periods for API event and summary data to suit your requirements and available storage space.

Shards

To efficiently distribute the data in OpenSearch indices across a cluster, the indices are subdivided into what are called "shards", which can be stored in different nodes of the cluster.

Replicas

In a three node OpenSearch deployment, each shard (known as the "primary shard") has 2 replica shards that contain a copy of the data that is in the primary shard. OpenSearch stores the primary shard and each replica on different nodes to ensure that all data is available on all nodes, and to provide high-availability if one node fails.

A single node OpenSearch deployment has no replica shards, since no other nodes are available to store them.

Installing the analytics subsystem

Links to the installation steps for each platform.

For all features and settings that must be decided and configured before installation, see Preinstallation planning.

It is recommended configure your backup settings after installation. See backup and disaster recovery.

Follow the installation steps for your platform:

- Kubernetes
- OpenShift and Cloud Pak for Integration
- VMware one-replica deployment
- VMware three-replica deployment

Backup, restore, and disaster recovery

How to back up your analytics subsystem and restore after a disaster.

An analytics subsystem backup consists of:

Analytics infrastructure configuration backup
: Your infrastructure configuration backup contains information such as endpoints, offload configuration, remote backup site credentials.
The infrastructure configuration backup is also referred to as the "config backup".

  On VMware deployments, the config backup includes everything that you configure with the **apicup** command, and typically consists of the contents of your project directory. On Kubernetes and OpenShift, the config backup consists of YAML file backups of your analytics CR and certificates.

  Your config backup is needed for disaster recovery, where you must redeploy your analytics subsystem.

Analytics database backup
: Your database backup contains all your API event data, summary data, and shared queries. Database backups are automatically created and stored on an S3 object store. The configuration of your database backups, such as S3 object store credentials, are defined in your analytics infrastructure configuration. Therefore, to restore an analytics database backup after a disaster, you must first restore your config backup.
Your analytics database backups are intended for use in two scenarios:

  - Disaster recovery: Your entire analytics subsystem fails. You redeploy the subsystem (use your config backup to ensure that the same config settings and certificates are defined) and then restore your most recent database backup.
  - Corrupted index recovery: One or more analytics indices are corrupted, so you restore just those indices from a backup that contains them.

  Analytics database backups are not intended for long-term storage of your analytics data. If you require long-term storage, use the offload or export features.
  Note: When you restore a database backup, any indices in the backup that are older than your retention period, are restored. However, the daily retention and rollover job will delete these old indices the next time that it runs.

Review the backup, restore, and disaster recovery documentation for your platform:

- **Kubernetes, OpenShift, and Cloud Pak for Integration**
  How to configure analytics backups and recover from a disaster in Kubernetes, OpenShift, and Cloud Pak for Integration environments.
- **VMware**
  How to configure analytics backups and recover from a disaster in a VMware environment.

Kubernetes, OpenShift, and Cloud Pak for Integration

How to configure analytics backups and recover from a disaster in Kubernetes, OpenShift, and Cloud Pak for Integration environments.

The analytics subsystem backup consists of:

- Analytics infrastructure configuration backup:
  - Backup of your analytics CR YAML file.
  - Backup of any custom certificates you configured. For example, the certificates that secure your communication with your remote object-store, and any certificates used for connecting to third-party systems for analytics offload.

  The infrastructure configuration backup is also referred to as the "config backup".

  If you have a disaster, your config backup is essential for restoring your analytics subsystem.

- Analytics database backup. Analytics database backups must be stored on an S3 compatible object-store, for example, IBM Cloud Object Storage.
  Note:
  - Only host-style S3 backups are supported. Path-style S3 backups are not supported.
  - If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.

Analytics subsystem backups are intended for restoration in the same environment where they were taken, or a different environment that has the same network configuration, endpoints, and deployment profile.

- **Taking an analytics infrastructure configuration backup**
  How to backup your analytics subsystem deployment configuration.
- **Configuring analytics database backups**
  Configure scheduled database backups of your analytics subsystem.
- **Running an on-demand analytics database backup**
  Deploy the `AnalyticsBackup` CR to initiate a backup of the analytics database.
- **Restoring the analytics database**
  Deploy the `AnalyticsRestore` CR to initiate restoration of an analytics database backup.
- **Recovering from a disaster**
  Reinstall your analytics subsystem, restore your analytics CR, secrets, and your database backup.
- **Deleting old backups**
  Manage the retention of your analytics backups.

- **Updating your S3 object store credentials**
  Update the secret that secures your analytics backups in your S3 object store.
- **Troubleshooting backup and restore**
  Troubleshooting steps for problems with backup and restore of the analytics database.

Taking an analytics infrastructure configuration backup

How to backup your analytics subsystem deployment configuration.

About this task

The analytics infrastructure configuration backup consists of:

- Backup of your analytics CR YAML file.
- Backup of any custom certificates you configured. For example, the certificates that secure your communication with your remote S3 object-store, and any certificates used for connecting to third-party systems for analytics offload.

Note: Every time that you update the analytics subsystem CR or any custom analytics certificates, take a new analytics infrastructure configuration backup.
The infrastructure configuration backup is also referred to as the "config backup".

If you have a disaster, your config backup is essential for restoring your analytics subsystem.

Note: For OpenShift users: The steps that are detailed in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place.

Procedure

1. Backup your analytics CR to a file called analytics-subsystem-cr-backup.yaml:

   ```
   kubectl get a7s -o yaml > analytics-subsystem-cr-backup.yaml
   ```

2. Backup your analytics database backup secret to a file called analytics-backup-secret_backup.yaml:

   ```
   kubectl get secret analytics-backup-secret -o yaml > analytics-backup-secret_backup.yaml
   ```

   The backup secret is created when you configure analytics database backups: Configuring analytics database backups. If database backups are not yet configured, ensure that this secret is backed up when it is configured.

3. If you are using the analytics offload feature, then backup any custom offload certificates that are used:

   ```
   kubectl get secret <offload secret name> -o yaml > analytics-offload-secret_backup.yaml
   ```

   For more information about offload certificates, see Offload to third-party systems.

4. If your S3 object store has a self-signed CA certificate, then backup the secret that contains this certificate:

   ```
   kubectl get secret <S3 CA secret name> -o yaml > analytics-S3-CA-secret_backup.yaml
   ```

   For more information about S3 CA certificates, see Creating the S3 CA secret.

5. If you manually created or customized any other analytics subsystem certificates, then take a backup of them:
   To identify all the secrets in your analytics namespace, run:

   ```
   kubectl get secrets
   ```

   For each secret that you want to backup, use the following command:

   ```
   kubectl get secret <secret name> -o yaml > <backup filename>
   ```

   You do not need to backup any secrets that cert-manager created and maintains. For more information about analytics TLS certificates, see API Connect TLS certificates.

6. Copy all the backup files that were created in the previous steps to a safe location.

Configuring analytics database backups

Configure scheduled database backups of your analytics subsystem.

Procedure

1. Create the analytics backup secret for your S3 object-store.
   a. Follow the documentation of your S3 provider to obtain an access key and secret. Documentation links for some common S3 providers:
      - IBM Cloud Object Storage
      - AWS
   b. Create a Kubernetes secret called `analytics-backup-secret` that contains your access key and secret, in the same namespace as your analytics subsystem:

      ```
      kubectl -n <namespace> create secret generic analytics-backup-secret --from-literal=access_key='<your access key>' --from-literal=secret_key='<your access secret>'
      ```

Note: If you change your access key and secret, then you must create a new secret that has a different name. For more information about updating your backup secret, see Updating your S3 object store credentials.

2. Open the analytics CR for editing.
   On Kubernetes and OpenShift individual subsystem installations, edit your analytics CR directly:

   ```
   kubectl edit a7s
   ```

   On Cloud Pak for Integration and OpenShift top-level CR installations, edit the `APIConnectCluster` CR:

   ```
   kubectl edit APIConnectCluster
   ```

3. Add the `databaseBackup` configuration.
   On Kubernetes and OpenShift individual subsystem installations, add `databaseBackup` to the `spec` section of the analytics CR. On Cloud Pak for Integration and OpenShift top-level CR installations, add `databaseBackup` to the `spec.analytics` section of the `APIConnectCluster` CR.

   ```
   databaseBackup:
     credentials: analytics-backup-secret
     host: <S3 host endpoint>
     path: <S3 path>
     backupCerts: <S3 server CA secret>
     schedule: <cron formatted schedule> # "0 2 * * *"
     enableCompression: true
     chunkSize: 1gb
     enableServerSideEncryption: false
   ```

   where:

   - *<S3 host endpoint>* is the S3 endpoint with the corresponding S3 region. Examples:
     - Region provided in the FQDN:

       `s3.s3region.s3domain`

     - Region provided in the path:

       `s3.s3domain/s3region`

     - If the endpoint has no region:

       `s3.s3domain`

     Note: A path-style host value is not supported for AWS:

     `s3.region-code.amazonaws.com/bucket-name`

     In the preceding example, the *bucket-name* is interpreted as the region. With AWS, use only host style, where the bucket-name is in the hostname:

     `bucket-name.s3.amazonaws.com/region-code`

   - *<S3 path>* is a combination of the S3 bucket and the base path within the bucket. Use the format: `bucket_name/path`
   - *<S3 server CA secret>* is the name of a secret that contains your S3 server certificate. This property is only required if your S3 server has a self-signed root certificate, see Creating the S3 CA secret. If your S3 server does not have a self-signed root certificate then you can delete this line.
   - *<cron formatted schedule>* is the schedule that determines how often backups are run. The format for the schedule is any valid `cron` string. The time zone for backups is that of the `kube-controller-manager`.
     Tip: To temporarily disable scheduled backups, but keep the rest of your backup configuration, comment out the `schedule` property.
   - Optional: `enableCompression` determines whether metadata files are stored in compressed format. The default value is true.
   - Optional: `chunkSize` specifies how files are stored. Large files can be stored as chunks when the snapshot is created. This setting specifies the size of the chunks. Format is lowercase and no spaces, for example: `1gb`, `50mb`, `800kb`.
   - Optional: `enableServerSideEncryption` determines whether files are encrypted. When set to true, files are encrypted on the server side with AES256. The default value is false.

What to do next

After your backup settings are configured in your analytics CR, take a new config backup to capture the updated CR settings and backup secrets.

- **Updating the analytics database backup secret**
  Update the secret that secures your analytics database backup.

Creating the S3 CA secret

Configure your analytics subsystem to trust your S3 object-store.

If your S3 object store presents a self-signed public server certificate, create a Kubernetes secret that contains the certificate, and specify the secret in your analytics CR.

1. Create a file called `ca.crt` that contains your S3 object store public server certificate.
   Note: If the certificate includes intermediate certificates, then provide the entire chain in the `ca.crt` file, for example:

   ```
   intermediate-certificate-1
   intermediate-certificate-2
   . . .
   intermediate-certificate-n
   root-certificate
   ```

   where the first certificate in the `ca.crt` (intermediate-certificate-1) is the issuer of the S3/object-store server certificate, and each subsequent certificate is the issuer of the preceding certificate.
2. Encode your certificate in base64 and remove any newline characters:

```
base64 <path to ca.crt file> | tr -d '\n'
```

3. Create a file that is called custom_s3_ca.yaml and paste in the following text:

```
apiVersion: v1
data:
  ca.crt: <base64 encoded certificate>
kind: Secret
metadata:
  name: custom-s3-server-ca
type: generic
```

   where *<base64 encoded certificate>* is the encoded string from step 2.
4. Apply the custom_s3_ca.yaml file to create your Kubenetes secret:

```
kubectl -n <namespace> apply -f custom_s3_ca.yaml
```

5. Specify your new secret in your analytics CR:

```
  databaseBackup:
    ...
    backupCerts: custom-s3-server-ca
    ...
```

Note: Remember to keep a backup of your S3 secret: Taking an analytics infrastructure configuration backup.

Updating the analytics database backup secret

Update the secret that secures your analytics database backup.

About this task

When you configure S3 storage for backing up analytics data, you encrypt the S3 provider's access key and save it as a Kubernetes secret for your analytics deployment. If the S3 access key or your Kubernetes secret is compromised, obtain a new key from the S3 provider and use it to generate a new Kubernetes secret.

Important: When you invalidate a key, be sure to save a copy of it with a notation listing the dates when it was used. If you need to restore a backup that was created with the invalidated key, you will need to use that key to access the data. One method of saving the key is to create a new version of the CR every time you change the key, so that you can easily refer to the settings and key that were used for each backup.
This task assumes that you have already configured the backup settings for your analytics subsystem and now want to replace the Kubernetes secret that is specified as the **analytics-backup-secret** in the analytics CR.

Procedure

1. Update the secret for your S3 storage:
   a. Obtain a new access key and corresponding access key secret from your S3 provider.
   b. Invalidate the previous credentials that were specified in the analytics CR.
2. Delete the invalidated Kubernetes secret by running the following command:

```
kubectl -n namespace delete secret name
```

   where:

   - ***namespace*** is the namespace where analytics is deployed.
   - ***name*** is the name of the current (invalidated) Kubernetes secret, which is specified in the **credentials** setting in the **databaseBackup** section of the analytics CR. Every time you update the secret, you must assign it a new name.
3. Create a new Kubernetes secret by running the following command and filling in your access key, access key secret, and namespace:

```
kubectl -n namespace create secret generic new_secret_name --from-literal=access_key='S3_access_key' --from-
literal=secret_key='S3_access_key_secret'
```

   where:

   - ***namespace*** is the namespace where analytics is deployed.
   - ***new_secret_name*** is a new name for storing the new Kubernetes secret. Every time you update the secret, you must assign it a new name.
   - ***S3_access_key*** and ***S3_access_key_secret*** are the values that you received from the S3 provider.
4. Edit the analytics CR file and replace the invalidated secret with the new secret.
   a. In the CR, locate the **credentials** setting in the **databaseBackup** section.
   b. Replace the invalidated secret name with the name of the new secret that you just generated.
   c. Save and close the CR file.
5. Run the following command to apply the updated CR so that the new secret takes effect:

```
kubectl -n namespace apply -f path/to/analytics-cr
```

   where:

   - ***namespace*** is the namespace where analytics is deployed.
   - ***path/to/analytics-cr*** is the path and file name of the analytics CR.

Running an on-demand analytics database backup

Deploy the **AnalyticsBackup** CR to initiate a backup of the analytics database.

Before you begin

Configure your backup settings in your analytics CR: Configuring analytics database backups.

About this task

An on-demand backup is triggered by creating an **AnalyticsBackup** CR. The API Connect operator detects the presence of the backup CR and runs the backup job.

Note: You can run only one backup or restore process at a time.

Procedure

1. Create a file called analyticsbackup_cr.yaml and paste in the following contents:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  generateName: <backup prefix>
spec:
  name: <backup name>
  crType: create
  comment: <comment>
```

   where:
   - *<backup prefix>* is prefix for the name of your backup. The remainder of the name is automatically generated. Keep the prefix short, 8 characters or less.
   - Optional: *<backup name>*. If you don't want an autogenerated backup name with a prefix, you can specify the complete backup name here and remove the **metadata.generateName** property.
   - *<comment>* is an optional comment that you can give your backup, for example "load test backup".

   Advanced settings in this CR that you can configure are:
   - **indices:** specify the indices that you want to include in the backup. Options are: **all|apievents|ui|summary**, or the names of individual indices. **all** includes **apievents**, **ui**, and **summary** indices.
   - **enableIgnoreUnavailable:** set to **true** if you want the backup to continue even if some of the indices you specified in the **indices:** property are not found in your analytics database.
2. Deploy your CR by running the following command to create it in the namespace of your analytics subsystem:

```
kubectl -n <namespace> create -f analyticsbackup_cr.yaml
```

3. To see the progress of your backup, run the following command:

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

```
NAME                  STATUS   ID                                     CR TYPE   INDICES          AGE   COMMENT
<backup name>         Running                                         create    [all]            12s   <comment>
```

   When the backup is complete the status changes from **Running** to **Ready**.
4. Display the details for your backup by running the following command:

```
kubectl  -n <namespace> get analyticsbackup <backup name> -o yaml
```

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsBackup
metadata:
  creationTimestamp: "2020-05-20T16:18:05Z"
  generateName: a7s-bup-
  generation: 1
  name: a7s-bup-gb6s5
  namespace: my-namespace
  resourceVersion: "867329"
  uid: f43b8400-733d-411a-9e9a-de2166bbfb0a
spec:
  indices:
  - all
status:
  conditions:
  - lastTransitionTime: "2022-06-30T16:18:24Z"
    message: SUCCESS
    status: Complete
    type: Ready
  details: |
    {Name:my-backup-id
    Version:5.6.16
    Indices:[...]
    StartTime:2022-06-30T16:18:09.698Z
    EndTime:2022-06-30T16:18:16.082Z
    State:SUCCESS
    Failures:[]
    Shards:{Failed:0
    Successful:1}}
  id: my-backup-id
```

Restoring the analytics database

Deploy the **AnalyticsRestore** CR to initiate restoration of an analytics database backup.

Before you begin

Analytics database backups are intended for restoration of the complete analytics database after a disaster, or the recovery of specific analytics indices that are identified as corrupted on your existing deployment. If you are restoring from backup to recover a specific index or indices, ensure that you are familiar with the OpenSearch concepts.
Note: The analytics database uses time-based indexing so that each day of API event data has its own analytics index. Before your restore a backup, understand the following points:

- The analytics database backup consists of separate backups of each OpenSearch index. You can restore all indices in the backups, or just some of them.
- The restore process does not touch existing indices that are not in the backup. So, if you restore an analytics backup that you took yesterday, your API event data from today remains because your backup contains no API event index from today.
- The restore process does not by default overwrite analytics indices that exist in the backup and in your current analytics database. You can force the restore to overwrite existing indices with **enableOverride**.
- Indices in the backup that are older than your retention period are restored, but are later deleted when the daily retention and rollover job runs. Check your available persistent storage space before you attempt to restore an old backup. For example, if you have a retention period of 30 days, and your storage is 80% used, if you attempt to restore a backup that is 10 days old it is likely that the restore uses up the rest or your storage, and fails. Filling analytics persistent storage to 100% can lead to corrupted analytics indices and an analytics subsystem outage.
- If you are restoring an analytics backup that you took today, and you set **enableOverride** to true, any API event data that was generated since the backup was taken is lost.

About this task

Analytics database backups are restored by creating an **AnalyticsRestore** CR. The API Connect operator detects the presence of the restore CR, and triggers a restore of the backup that is specified in the CR.

Note: You can run only one backup or restore process at a time.

Procedure

1. Get a list of available backups by running the following command:

   ```
   kubectl -n namespace get analyticsbackup | grep record
   ```

   ```
   NAME                  STATUS   ID                                        CR TYPE   INDICES                AGE   COMMENT
   analytics-bup-kxwq5   Ready    analytics-all-2020-09-07t03:49:15utc   record    [all]                  34h
   analytics-bup-p29z8   Ready    analytics-all-2020-09-07t18:04:03utc   record    [all]                  20h
   ```

   Note the ID of the backup that you want to restore.
   Note: You cannot restore backups that have **CR TYPE** of **create**.
2. Create a file called analyticsrestore_cr.yaml and paste in the following text:

   ```
   apiVersion: analytics.apiconnect.ibm.com/v1beta1
   kind: AnalyticsRestore
   metadata:
     generateName: a7s-restore-
   spec:
     indices:
     - all
     backupID: <backup ID>
     enableIgnoreUnavailable: false
     enableOverride: true
   ```

   set *<backup ID>* to the name of the backup that you want to restore, identified in step 1.
   Note: The restore CR has some advanced properties that you can set. The advanced properties are:
   - **generateName** is the prefix for the name of the restore instance that is created.
   - **indices** is a list of index names or keywords, indicating which indices you want to restore. Options are: **all|apievents|ui|summary**, or the names of individual indices. **all** includes **apievents**, **ui**, and **summary** indices. If no indices are specified, then all indices in the backup are restored. For more information about selecting individual indices for restore, see Identifying specific indices to restore.
   - **enableIgnoreUnavailable** specifies whether the restore aborts if an index that is listed in the **indices** property is not found. If set to **true** any missing indices are ignored and the restore continues.
   - **enableOverride** specifies if the restore should overwrite any existing indices that you have in your analytics database. Set **enableOverride** to **false** if you want the restore to fail if it finds you already have the same indices in your current analytics database.
3. Deploy your CR by running the following command to create it in the namespace of your analytics subsystem:

   ```
   kubectl -n <namespace> create -f analyticsrestore_cr.yaml
   ```

4. To monitor your restore, run the following command:

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

   ```
   NAME               STATUS     ID                                        AGE
   a7s-restore-9z4fr Ready      analytics-all-2022-07-01t04:49:15utc              1h
   ```

   When the restore is successfully started, **STATUS** shows **Running**. The status that is shown for completed restores depends on your deployment profile:
   - On a three replica deployment, **STATUS** shows **Complete**.
   - On a one replica deployment, **STATUS** shows **Warning**. Review step 5 for information on checking restore status.
5. Display the details for a particular restore by running the following command:

   ```
   kubectl -n <namespace> get analyticsrestore <restore name> -o yaml
   ```

   where *<restore name>* is the name of your restore, that you identified in step 4.
   Example output for a one replica deployment where the restore is complete:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  creationTimestamp: "2022-07-01T04:06:03Z"
  generateName: a7s-restore-
  generation: 1
  name: a7s-restore-9z4fr
  namespace: my-namespace
  resourceVersion: "975152"
  uid: fdda9105-17ab-44ac-83de-852620a4dcde
spec:
  backupID: analytics-all-2022-07-01t03:49:15utc
  enableIgnoreUnavailable: false
  enableOverride: true
  indices:
  - all
status:
  conditions:
  - lastTransitionTime: "2022-07-01T04:08:03Z"
    message: WARNING
    status: Warning
    type: Ready
  details: |
    &{NumberOfDataNodes:1
    Status:yellow
    ActivePrimaryShards:23
    ActiveShards:23
    RelocatingShards:0
    InitializingShards:0
    UnassignedShards:40
    PendingTasks:0}
  id: analytics-all-2022-07-01t04:49:15utc
```

The **Warning** status can be ignored on a one replica deployment, it is due to there not being other nodes to store replica shards on. Restore is complete when **status.details.Status** is yellow, and **status.details.InitializingShards** is zero.

6. When restore is complete, delete the **AnalyticsRestore** CR:

   ```
   kubectl -n <namespace> delete analyticsrestore <restore name>
   ```

   Note: If your restore failed, and you want to investigate why, do not delete the **AnalyticsRestore** CR, it is needed for problem analysis.

Identifying specific indices to restore

How to identify specific indices and restore only those indices from an analytics database backup.

Because the analytics database stores analytics data in separate indices for each day, database corruption is often limited to just a few indices. You can save time on an analytics database restore by just restoring the corrupted indices. To understand more about how analytics data storage works, see OpenSearch concepts.
Note: Only follow the steps in this topic if IBM has identified that you have corrupted analytics indices. OpenSearch indices can sometimes show a status of "Red" when a communication problem between OpenSearch nodes occurs, and when the communication problem is resolved the status returns to "Yellow" or "Green".
To identify corrupted indices in your analytics subsystem, use the toolkit CLI:

1. Setup the toolkit CLI so you can run analytics CLI operations: Analytics CLI.
2. Get a list of your analytics indices with the **clustermgmt:catIndices** command:

   ```
   apic -m analytics clustermgmt:catIndices --analytics-service "<analytics service name>" -s <platform api endpoint> --
   format=json
   ```

   ```
   {
       "data": [
           {
               "health": "red",
               "status": "open",
               "index": "apic-api-2023.06.21-000007",
               "uuid": "pCaolBNoRWivgzGvv1kvLQ",
   ...
   ```

   corrupted indices have a health status of "red". The index property is the name of the index, in this example, the index name is "apic-api-2023.06.21-000007".

To restore indices from your analytics database backup, follow these steps:

1. Get a list of available backups by running the following command:

   ```
   kubectl -n namespace get analyticsbackup | grep record
   ```

   ```
   NAME                  STATUS   ID                                        CR TYPE   INDICES                AGE   COMMENT
   analytics-bup-kxwq5   Ready    analytics-all-2020-09-07t03:49:15utc   record    [all]                  34h
   analytics-bup-p29z8   Ready    analytics-all-2020-09-07t18:04:03utc   record    [all]                  20h
   ```

   Note the name and ID of the backup that you want to restore.
2. Describe the backup that you want to restore to see the list of indices in the backup:

   ```
   kubectl -n namespace describe analyticsbackup <backup name>
   ```

   where *<backup name>* is the name of the backup you identified in step 1.
   The indices in the backup are listed in the describe output:

   ```
   Indices:[apic-api-2023.06.21-000007
   apic-api-2023.06.30-000016
   ...
   ```

```
apic-api-2023.06.25-000011
apic-query-000001]
```

3. Follow the restore steps, but use the `indices` property to specify the indices that you want to restore, and set `enableOverride` to true so that any existing indices you have of the same name are overwritten, for example:

```
apiVersion: analytics.apiconnect.ibm.com/v1beta1
kind: AnalyticsRestore
metadata:
  generateName: a7s-restore-
spec:
  indices: apic-api-2023.06.30-000016,apic-api-2023.06.25-000011
  enableOverride: true
...
```

Note: The wildcard character * is supported in the indices list, so to restore all indices from the first 9 days of June, you can use:

```
indices: apic-api-2023.06.0*
```

Recovering from a disaster

Reinstall your analytics subsystem, restore your analytics CR, secrets, and your database backup.

About this task

Analytics subsystem backups can be restored in the same environment where they were taken, or in a different environment that has the same network configuration, endpoints, and deployment profile. You cannot restore an analytics subsystem backup in a different network environment, form factor, or deployment profile. If you want to move your API Connect deployment to a different form factor or change endpoints, see Migrating from v10 to v10 on a different form factor.

Procedure

1. Install the replacement analytics subsystem, configuring the same `ai-endpoint` as your original installation.
   Follow the installation procedures for your platform:
   - Kubernetes
   - OpenShift and Cloud Pak for Integration

   Refer to your infrastructure configuration backup to check the `ai-endpoint`. Also, if you disabled local storage, you must configure this in your analytics CR before installation.

2. Update the analytics CR on your replacement installation with the analytics backup settings and any other analytics features you configured in your previous installation.
   Your previous analytics settings are in your infrastructure configuration backup. If you are installed on Cloud Pak for Integration or OpenShift with the top-level CR, then add these settings to the `spec.analytics` section of your `APIConnectCluster` CR.
3. Re-create the analytics secrets that you backed up in your infrastructure configuration backup:
   a. Create the backup credential secret from your backup file: analytics-backup-secret_backup.yaml.

   ```
   kubectl -n <analytics namespace> apply -f analytics-backup-secret_backup.yaml
   ```

   b. If you are using the analytics offload feature, then create the offload secret from your backup file: analytics-offload-secret_backup.yaml.

   ```
   kubectl -n <analytics namespace> apply -f analytics-offload-secret_backup.yaml
   ```

   c. If your S3 object store has a self-signed CA certificate, then create the secret that contains the certificate from your backup file.

   ```
   kubectl -n <analytics namespace> apply -f analytics-S3-CA-secret_backup.yaml
   ```

   d. If you manually created or customized any other analytics subsystem certificates, then restore those secrets from their backup files.

   ```
   kubectl -n <analytics namespace> apply -f <backup filename>
   ```

4. Restore the analytics database: Restoring the analytics database.

Deleting old backups

Manage the retention of your analytics backups.

Warning: Do not delete backups directly from the active backup path of your S3 object store. Your analytics backups are incremental so deleting any of the files in the backup path can break all backups in that path.
To delete backups, change your S3 backup path to a new path, then delete the backup files from the old path:

1. Edit your `AnalyticsCluster` CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

   If you are installed on Cloud Pak for Integration or OpenShift with the top-level CR then edit the `spec.analytics` section of your `APIConnectCluster` CR:

   ```
   kubectl -n <namespace> edit APIConnectCluster
   ```

2. Update your backup path:

   ```
     databaseBackup:
   ```

```
...
    path: <S3 path>
...
```

<S3 path> is a combination of the S3 bucket and the base path within the bucket. Use the format: **bucket_name/path**.

3. Save and exit.
4. The next scheduled backup is stored in the new S3 path.
5. Delete old backups from your original S3 path.

Updating your S3 object store credentials

Update the secret that secures your analytics backups in your S3 object store.

About this task

When you configure you analytics backups, you create a Kubernetes secret that contains credentials for your S3 object store. If your S3 credentials change, or you want to invalidate your existing credentials and create new ones, then follow the steps in this procedure.

This task assumes that you have already configured analytics backups and want to replace the **analytics-backup-secret** that is specified in the analytics CR.

Procedure

1. Refer to your S3 provider's documentation to obtain new access credentials and invalidate your previous ones.
2. Delete your existing analytics backup secret:

   ```
   kubectl -n <namespace> delete secret <secret name>
   ```

   where:
   - *<namespace>* is you analytics namespace.
   - *<secret name>* is the name of you existing secret, which is specified in the **databaseBackup.credentials** section of your analytics CR:

     ```
     spec:
       databaseBackup:
         credentials: analytics-backup-secret
     ...
     ```

     Note: Do not reuse the secret name for your replacement secret.
3. Create a new analytics backup secret:

   ```
   kubectl -n <namespace> create secret generic <new secret name> --from-literal=access_key='<S3 access key>' --from-
   literal=secret_key='<S3 access key secret>'
   ```

   where:
   - *<namespace>* is your analytics namespace.
   - *<new secret name>* is a new name for your analytics backup secret. Do not use the same name as your previous secret.
   - *<S3 access key>* and *<S3 access key secret>* are provided by your S3 provider.
4. **Kubernetes and OpenShift individual subsystem installations:** Edit your analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

   Your backup settings are in the **spec:** section.
5. **Cloud Pak for Integration and OpenShift top-level CR installations:** Edit your **APIConnectCluster** CR:

   ```
   kubectl -n <namespace> edit APIConnectCluster
   ```

   Your backup settings are in the **spec.analytics** section.
6. Replace the **databaseBackup.credentials** with your new secret:

   ```
     databaseBackup:
       credentials: <new analytics backup secret>
   ...
   ```

   After you save the updated CR, your analytics pods restart automatically.

Troubleshooting backup and restore

Troubleshooting steps for problems with backup and restore of the analytics database.

Check the CR status

List the analytics backup jobs:

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

```
NAME                  STATUS   ID                                    CR TYPE   INDICES                AGE   COMMENT
<backup name>         Ready                                          create    [all]                  <age>   <comment>
```

check the status of the backup that you are interested in:

```
kubectl -n <namespace> get analyticsbackup <backup name> -o yaml
```

List the analytics restore jobs:

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

```
NAME              STATUS     ID                                          AGE
<restore name>    Ready      analytics-all-2022-07-01t04:49:15utc              1h
```

check the status of the restore job you are interested in:

```
kubectl -n <namespace> get analyticsrestore <restore name> -o yaml
```

Check the logs

Check for backup or restore related errors in the logs of the API Connect operator pod:

1. Identify the API Connect operator pod `ibm-apiconnect` by running the following command:

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

   Note: *<namespace>* might not be the same namespace as your analytics subsystem. On a default OpenShift deployment, the API Connect operator runs in a global namespace called `openshift-operators`.
2. View the API Connect operator pod logs with:

   ```
   kubectl -n <namespace> logs <ibm-apiconnect pod name>
   ```

Check for messages with the "error" label that contain the words "analytics", "backup", or "restore".

Common problems with backup and restore

These are some frequent issues that can cause backup or restore to fail:

- Invalid login credentials for your remote backup server. Check that the username and password that you configured in your subsystem database backup settings is correct.
- The user does not have write permissions on the backup server. Check that the username you configured in your subsystem database backup settings has write permission to the specified backup path.
- Remote backup server storage full. Check your remote backup server and if the storage is full either extend the storage space or delete older backups.
- No network access to the remote backup server. Check that you can communicate with your remote backup server from your API Connect environment.
- TLS handshaking failure with the remote backup server. If your remote backup server has a self-signed CA certificate, check that this certificate is trusted by your API Connect deployment, see the database backup configuration steps for your subsystem for more information.

VMware

How to configure analytics backups and recover from a disaster in a VMware environment.

Analytics subsystem backups are intended for restoration in the same environment where they were taken, or a different environment that has the same network configuration, endpoints, and deployment profile.

- **Backing up your analytics subsystem**
  Backup your project directory and configure scheduled database backups.
- **Running an on-demand database backup**
  Run a backup of the analytics database outside the scheduled time.
- **Restoring an analytics database backup**
  Restore an analytics database backup.
- **Recovering from a disaster**
  Recover the analytics subsystem in a VMware environment.

Backing up your analytics subsystem

Backup your project directory and configure scheduled database backups.

Project directory backups

The original project directory that was created with the `apicup` command during the initial product installation is essential for disaster recovery, see: First steps for deploying in a VMware environment. The project directory contains the YAML files that describe your deployment, encryption keys, and deployment ISO files. It is not possible to restore the analytics database on a deployment that uses a new or different project directory.

Take a new backup of your project directory every time you make any configuration changes with the `apicup` command. Store your project directory backups at a remote location.

Ensure that a backup is kept of your Analytics extra-values file, if this file is not located in your project directory.

Analytics database backups

The analytics database backups require a remote S3 object store; for example, IBM Cloud Object Storage.
Note:

- Only host-style S3 backup endpoint names are supported. Path-style names are not supported.
- If you are using a self-managed S3 backup server, check that you have a DNS hostname entry and matching TLS certificate for your S3 backup endpoint.

Use the **apicup** command to configure backup settings for the analytics database.

1. Configure the backup settings with the **apicup** command:

   ```
   apicup subsys set <analytics subsystem> <key> <value>
   ```

   The configurable backup settings are:

   Table 1. Analytics backup settings

Key	Default value	Description
`analytics-backup-auth-pass`		S3 object store password.
`analytics-backup-auth-user`		S3 object store username.

`analytics-backup-host`		The S3 endpoint with the corresponding S3 region. Examples:
		• Region provided in the FQDN:
		`s3.s3region.s3domain`
		• Region provided in the path:
		`s3.s3domain/s3region`
		• If the endpoint has no region:
		`s3.s3domain`
		Note: A path-style host value is not supported for AWS:
		`s3.region-code.amazonaws.com/bucket-name`
		In the preceding example, the *bucket-name* is interpreted as the region. With AWS, use only host style, where the bucket-name is in the hostname:
		`bucket-name.s3.amazonaws.com/region-code`

`analytics-backup-certs`		The name of the Kubernetes secret inside your analytics subsystem that contains the S3 CA certificate. Follow the steps on Creating the S3 CA secret to create and assign this secret.
`analytics-backup-path`		A combination of the S3 bucket and the base path within the bucket, with the format: `bucket_name/path`
`analytics-backup-chunk-size`	1gb	(Optional) Specifies the size of large file chunks. Format is lowercase and no spaces, for example: `1gb`, `50mb`, `800kb`.
`analytics-backup-schedule`	0 2 * * *	(Optional) Determines how often backups are run. The format for the schedule is any valid `cron` string. The time zone for backups is Coordinated Universal Time.
`analytics-enable-compression`	True	(Optional) Determines whether metadata files are stored in compressed format.
`analytics-enable-server-side-encryption`	False	(Optional) Determines whether files are encrypted. When set to true, files are encrypted on the server side with AES256.

2. Validate the backup settings:

   ```
   apicup subsys get <analytics subsystem> --validate
   ```

3. Apply the updated backup settings:

   ```
   apicup subsys install <analytics subsystem>
   ```

4. Run the health check to confirm that the subsystem handled the update successfully :

   ```
   apicup subsys health-check <analytics subsystem>
   ```

   No output is returned if no problems are detected.

Snapshots

Take a virtual machine (VM) snapshot of all your analytics VMs; see Using VM snapshots for infrastructure backup and disaster recovery. Taking snapshots does require a brief outage while all the VMs in the subsystem cluster are shut down - do not take snapshots of running VMs, as they might not restore successfully. VM snapshots can provide faster recovery when compared to redeploying OVAs and restoring from normal backups.

Important: VM snapshots are not an alternative to the standard API Connect backups that are described in the previous step. Do not rely solely on VM snapshots for your backups.

Creating the S3 CA secret

Configure your analytics subsystem to trust your S3 object store.

If your S3 object store presents a self-signed public server certificate, create a Kubernetes secret that contains the certificate, and specify the secret in your analytics CR.

1. Use an SSH client to login to your analytics VM as the `apicadm` user:

   ```
   ssh apicadm@<analytics VM FQDN>
   ```

   If you have a three replica deployment, you can login to any one of the VMs. You do not need to repeat these steps on all VMs.
2. Switch to the root user:

   ```
   sudo -i
   ```

3. Create a file called `ca.crt` that contains your S3 object store public server certificate:

   ```
   echo "<certificate>" > ca.crt
   ```

   where *<certificate>* contains the full chain of your S3 CA certificate, for example:

   ```
   intermediate-certificate-1
   intermediate-certificate-2
   . . .
   intermediate-certificate-n
   root-certificate
   ```

4. Encode your certificate in base64 and remove any newline characters:

   ```
   base64 ca.crt | tr -d '\n'
   ```

5. Create a file that is called custom_s3_ca.yaml and paste in the following text:

   ```
   apiVersion: v1
   data:
     ca.crt: <base64 encoded certificate>
   kind: Secret
   metadata:
     name: custom-s3-server-ca
   type: generic
   ```

   where *<base64 encoded certificate>* is the encoded string from step 4.
6. Apply the custom_s3_ca.yaml file to create your Kubenetes secret:

   ```
   kubectl apply -f custom_s3_ca.yaml
   ```

7. Verify that the secret is created:

   ```
   kubectl get secrets
   ```

   expected output is:

   ```
   NAME                                                      TYPE                  DATA   AGE
   ...
   custom-s3-server-ca                                       generic               1      7s
   ...
   ```

8. Take a backup of the new secret. Run:

   ```
   kubectl get secret custom-s3-server-ca -o yaml
   ```

   and copy the output to a new file in your project directory called custom-s3-server-ca.yaml.
9. Exit from your VM SSH session.
10. Set your `analytics-backup-certs` property to the Kubernetes secret that you created:

    ```
    apicup subsys set <analytics subsystem> analytics-backup-certs custom-s3-server-ca
    ```

Running an on-demand database backup

Run a backup of the analytics database outside the scheduled time.

About this task

You can take an analytics backup at any time, you do not need to wait for the scheduled database backups to run. However, avoid running multiple backups at the same time, or triggering an on-demand backup at the same time as the scheduled backup runs.

Procedure

1. Confirm that your analytics subsystem is working correctly:

   ```
   apicup subsys health-check <analytics subsystem>
   ```

   if successful, the command returns silently with exit code 0.
2. Start the backup by running the following command and including the appropriate flags:

   ```
   apicup subsys backup <analytics subsystem>
   ```

   Note: Run the backup command with `--help` to see the advanced backup options:

   ```
   Backup analytics subsystem

   Usage:
     backup SUBSYS [flags]

   Flags:
         --comment string          Comment to add to the backup CR before creation
         --debug                   Enable debug logging
     -h, --help                    help for backup
         --ignore-unavailable      If false, the backup will fail if an index is missing. If true, missing indices will be
   skipped and the backup will continue.
         --indices strings         Indice(s) you'd like to backup (all/apievents/ui/summary). It can be set to any
   combination of indices (e.g. ui or ui,apievents).
   ```

```
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

The `--indices` flag can be set to `all|apievents|ui|summary`, or the names of individual indexes. `all` includes `apievents`, `ui`, and `summary` indices. Multiple values can be supplied with a comma-separated list. For example, to backup all UI data and all data from August of 2022:

```
--indices ui, "apic-api-2022.08*"
```

If you provide a list of indices in the `--indices` property, then set the `--ignore-unavailable` property to define what happens if one or more of the indices in your list are not found in the backup.

3. Monitor the progress of your backup with:

```
apicup subsys list-backups <analytics subsystem>
```

```
NAME                  STATUS   ID                                          CR TYPE   INDICES              AGE   COMMENT
a7s-backup-bsbft      Ready                                                create    ["all"]              13m
analytics-bup-dcxqh   Ready    defanalytics-all-2023-06-29t09:29:53utc     record    ["apievents","ui"]   13m
```

When the backup is complete the status changes from `Running` to `Ready`.

Restoring an analytics database backup

Restore an analytics database backup.

Procedure

1. Retrieve a list of available backups that have the `CR TYPE` of `record`:

```
apicup subsys list-backups <analytics subsystem> | grep record
```

```
NAME                  STATUS   ID                                          CR TYPE   INDICES              AGE   COMMENT
analytics-bup-dcxqh   Ready    defanalytics-all-2023-06-29t09:29:53utc     record    ["apievents","ui"]   60m
```

The response includes the ID of each backup. Note the ID of the backup that you want to restore.

2. Restore the backup that you identified in step 1:

```
apicup subsys restore <analytics_subsystem> --id "<backup ID>"
```

Note: Run the restore command with `--help` to see the advanced restore options:

```
Restore analytics subsystem

Usage:
   restore SUBSYS [flags]

Flags:
      --debug                   Enable debug logging
  -h, --help                    help for restore
      --id string               ID of the backup to restore
      --ignore-unavailable      If false, the restore will fail if an index is missing. If true, missing indices will be
skipped and the restore will continue.
      --indices strings         Indice(s) you'd like to restore (all/apievents/ui). It can be set to any combination of
indice(s) (eg. ui or ui,apievents) (default [all])
      --override                If set to true, then the restore operation will override existing indices.
      --wait                    Wait for the operation to complete or fail.
      --wait-timeout duration   Command timeout in seconds. (default 40s)
```

- `--id`: the backup ID provided must be enclosed in double quotation marks.
- `--indices` can be set to `all|apievents|ui|summary`, or the names of individual indices. `all` includes `apievents`, `ui`, and `summary` indices. Multiple values can be supplied with a comma-separated list. For example, to restore all UI data and all data from August of 2022:

  ```
  --indices "ui,apic-api-2022.08*"
  ```

- `--ignore-unavailable`: If you provide a list of indices in the `--indices` property, then set the `--ignore-unavailable` property to define what happens if one or more of the indices in your list are not found in your backup.
- `--override`: Specifies whether any indices that already exist in your database are overwritten by indices of the same name from the backup. The default value is `false` to prevent you from accidentally replacing existing indices.

3. To monitor the progress of your restore:

```
apicup subsys list-restores <analytics subsystem>
```

```
NAME                STATUS    ID                                          AGE
a7s-restore-9bwfv   Running   defanalytics-all-2023-06-29t09:29:53utc     9s
```

When the restore is successfully started, `STATUS` shows `Running`. The status that is shown for completed restorations depends on your deployment profile:

- On a three replica deployment, `STATUS` shows `Complete`.
- On a one replica deployment, `STATUS` shows `Warning`. SSH into your analytics VM and review the Kubernetes platform restore steps to confirm restore status.

Recovering from a disaster

Recover the analytics subsystem in a VMware environment.

Recovering the analytics subsystem involves re-creating your analytics ISO files, redeploying you analytics VMs with the new ISO files, and then restoring the analytics database from your backup.

In a clustered deployment, if any one VM is corrupted then all the VMs in the cluster must be redeployed. You cannot replace just a single corrupted VM in a cluster.

Use the same project directory that you used for your original deployment, or a restore of your project directory backup.

1. Create your ISO files:

   ```
   apicup subsys install <install subsystem name> --out <output directory>
   ```

   The ISO files are created in the directory that is specified in *<output directory>*.
   Note: If your original ISO files are still available and you did not upgrade from the original installation, you can reuse them. However, if you upgraded your original deployment, you must create new ISO files by using the version of **apicup** that corresponds to the version your API Connect installation was on at the time of the disaster. For example, do not attempt to deploy v10.0.6.x OVAs with ISO files that were created with **apicup** v10.0.5.3.
2. Deploy your new analytics VMs. See Deploying the Analytics subsystem OVA file.
3. Verify your new analytics deployment. See Verifying deployment of the Analytics subsystem.
4. Restore your analytics database backup: Restoring an analytics database backup.

API event record field reference

Reference table and examples of the data that is stored in an API event record.

When a user calls one of your published APIs, the gateway that processes the API call creates an API event record. Each API event record contains details of the API call such as the application ID, current time, result code, etc. Table 1 shows all the data that can be contained in an API event record.

Activity log settings

The contents of API event records depends on the activity log setting for each API, and any custom gateway scripts and ingestion filters that are configured. To find out more about activity logging and customizing API event contents, see Customize your analytics data.

API event record fields

Table 1. API event record fields

Field name	Type	Description
@timestamp	Date	A timestamp that records when the Logstash data collection engine (which feeds data into OpenSearch) wrote the record.
api_id	String	The API identifier.
api_name	String	The name of the API.
api_resource_id	String	Field format is: **api_name:api_version:method:path**. Only available on API Gateway v10.5.3 or higher.
api_version	String	The version number of the API.
app_id	String	The identifier for the registered application.
app_name	String	The name of the registered application. Note: The property is set to **undefined** when a client ID is not used or is invalid on the API. The gateway needs a client ID to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app was invoked, because a single API can belong to multiple products (each of which have multiple plans and apps that are subscribed to those plans with client IDs).
app_type	String	The application type, with a value of **Production** or **Development**.
bytes_received	Number	The number of bytes received from the consumer in the request.
bytes_sent	Number	The number of bytes sent to the consumer in the response.
client_geoip.area_code	Number	The public switched telephone network (PSTN) area code of the client, as identified from its IP address.
client_geoip.asn	Number	Autonomous system number of client IP address.
client_geoip.as_org	String	
client_geoip.city_name	String	The city name of the client, as identified from its IP address.
client_geoip.continent_name	String	The continent name of the client, as identified from its IP address.
client_geoip.continent_code	String	The two-letter continent code of the client, as identified from its IP address.
client_geoip.country_code2	String	The two-letter country code of the client, as identified from its IP address.
client_geoip.country_code3	String	The three-letter country code of the client, as identified from its IP address.
client_geoip.country_name	String	The country name of the client, as identified from its IP address.
client_geoip.dma_code	Number	The Designated Market Area (DMA) code of the client, as identified from its IP address.
client_geoip.domain	String	The domain of the client IP address.
client_geoip.ip	String	The IP address of the client.
client_geoip.isp	String	The internet service provider of the client.

Field name	Type	Description
client_geoip.latitude	Number	The latitude of the client location, as identified from its IP address.
client_geoip.location	String	The longitude and latitude of the client location (separated by a comma), as identified from its IP address.
client_geoip.location.lat	Number	The latitude of the client location, as identified from its IP address.
client_geoip.location.long	Number	The longitude of the client location, as identified from its IP address.
client_geoip.longitude	Number	The longitude of the client location, as identified from its IP address.
client_geoip.organization	String	The client's organization, as identified from its IP address.
client_geoip.postal_code	String	The postal code of the client, as identified from its IP address.
client_geoip.region_iso_code	String	The ISO code of the client's region, as identified from its IP address.
client_geoip.region_name	String	The abbreviated form of the region that corresponds to the IP address of the client.
client_geoip.timezone	String	The time zone of the client, as identified from its IP address.
client_id	String	The unique ID of the client that is attached to the API request.
custom_data	Array map	Custom data can be added to this field.
datetime	Date	A timestamp that records when the API was invoked. The timestamp is always shown in Coordinated Universal Time.
developer_org_id	String	The identifier for the consumer organization that owns the application.
developer_org_name	String	The name of the consumer organization that owns the application.
endpoint_url	String	When the request failed, **`endpoint_url`** identifies the proxy or invoke target URL on which the request failed. It is not included with a successful request.
env_id	String	The catalog identifier.
env_name	String	The name of the catalog.
event_id	String	Unique ID assigned to the event by the gateway.
gateway_geoip.area_code	Number	The public switched telephone network (PSTN) area code of the gateway, as identified from its IP address.
gateway_geoip.asn	Number	Autonomous system number of client IP address.
gateway_geoip.as_org	String	
gateway_geoip.city_name	String	The city name of the gateway, as identified from its IP address.
gateway_geoip.continent_code	String	The two-letter continent code of the gateway, as identified from its IP address.
gateway_geoip.continent_name	String	The continent name of the gateway, as identified from its IP address.
gateway_geoip.country_code2	String	The two-letter country code of the gateway, as identified from its IP address.
gateway_geoip.country_code3	String	The three-letter country code of the gateway, as identified from its IP address.
gateway_geoip.country_name	String	The country name of the gateway, as identified from its IP address.
gateway_geoip.dma_code	Number	The Designated Market Area (DMA) code of the gateway, as identified from its IP address.
gateway_geoip.domain	String	The domain of the gateway IP address.
gateway_geoip.ip	String	The IP address of the gateway.
gateway_geoip.isp	String	The internet service provider of the gateway.
gateway_geoip.latitude	Number	The latitude of the gateway location, as identified from its IP address.
gateway_geoip.location	String	The longitude and latitude of the gateway location (separated by a comma), as identified from its IP address.
gateway_geoip.longitude	Number	The longitude of the gateway location, as identified from its IP address.
gateway_geoip.organization	String	The gateway's organization, as identified from its IP address.
gateway_geoip.postal_code	String	The postal code of the gateway, as identified from its IP address.
gateway_geoip.region_name	String	The abbreviated form of the region that corresponds to the IP address of the gateway.
gateway_geoip.region_iso_code	String	The ISO code of the gateway's region, as identified from its IP address.

Field name	Type	Description
gateway_geoip.timezone	String	The time zone of the gateway, as identified from its IP address.
gateway_ip	String	The IP address of the gateway.
gateway_service_name	String	The name of the API Connect gateway service. Configured by the cloud admin user when registering the gateway service. Only available on API Gateway v10.5.3 or higher.
global_transaction_id	String	The DataPower global transaction ID. See https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varserviceglobal-transaction-id-servicevarsglobaltransactionid.
graphql_request_field_cost	Number	GraphQL APIs only. The maximum cost of all fields accessed in the query. The cost of each field access is configured in the schema.
graphql_request_max_nesting	Number	GraphQL APIs only. The maximum nesting depth found in the query by the assembly validate action. The schema configuration is used to determine which types are nested, so this value might be less than the nesting depth found by the assembly parse action.
graphql_request_top_field_counts	Object	GraphQL APIs only. The maximum number of times that a query can retrieve each field. This number is equal to the number of times that the resolver is required to run. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_request_top_type_counts	Object	GraphQL APIs only. The maximum number of times that a query can retrieve an object of each type. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_request_type_cost	Number	GraphQL APIs only. The maximum cost of all types retrieved in the query. The cost of each type is configured in the schema.
graphql_response_field_cost	Number	GraphQL APIs only. The cost of all fields accessed in the query. The cost of each field access is configured in the schema.
graphql_response_max_nesting	Number	GraphQL APIs only. The nesting depth found in the query by the assembly validate action. The schema configuration is used to determine which types are considered nested, so this value might be less than the nesting depth found by the assembly parse action.
graphql_response_top_field_counts	Object	GraphQL APIs only. The number of times that each field was retrieved by the query. This number is equal to the number of times that the resolver is required to run. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_response_top_type_counts	Object	GraphQL APIs only. The number of times that an object of each type was retrieved by the query. This field is stored in JSON format and is not indexed, so it is not available for visualizations. A limited number of query requests and responses are stored for each entry, based on the amount of data that each contains. The maximum amount of data that can be stored is subject to change.
graphql_response_type_cost	Number	GraphQL APIs only. The cost of all types that were retrieved in the query. The cost of each type is configured in the schema.
headers.*field_name*	String	Internal information that is related to analytics ingestion. headers.*field_name* is not related to the API call or its response, see **`request_http_headers`** for that information.
host	String	The hostname or IP address of the ingestion node that received the API event.
http_user_agent	String	The value of the User Agent header on the inbound request.
immediate_client_ip	String	The client IP address that is directly in front of the gateway. Usually **`immediate_client_ip`** is the IP of a load balancer.
latency_info.started	Number	The time delay (in milliseconds) between when the request was received and when the gateway started the corresponding task. Starting a task comprises multiple steps to prepare for executing an API; for example, completing the TCP/TLS handshake, verifying an app's client ID and secret, and matching the request URI to a catalog, API, and Plan. When the gateway receives a request, the "Start" duration is set to 0. The duration of each step within the Start task is then added up, and the total represents the duration of the Start task.
latency_info.task	String	The API transaction that was processed.
log_policy	String	The defined logging policy. Values include none, event, headers, and payload.
org_id	String	The identifier for the provider organization that owns the API and associated Products.
org_name	String	The name of the provider organization that owns the API and associated Products.
plan_id	String	The Plan identifier.
plan_name	String	The name of the Plan. Note: The property is set to **`undefined`** when a client ID is not used or is invalid on the API. The gateway needs a client id to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app was invoked because a single API can belong to multiple products (each of which have multiple plans and apps that are subscribed to those plans with client IDs).
plan_version	String	The version number of the Plan.
product_name	String	The Product name. Note: The property is set to **`undefined`** when a client ID is not used or is invalid on the API. The gateway needs a client ID to determine which app was invoking the API. From this app, the gateway can determine what plan the app is subscribed to on the product that contains the API. Without a client ID, the gateway is unable to determine what plan, product, or app wasinvoked because a single API can belong to multiple products (each of which have multiple plans and apps that are subscribed to those plans with client IDs).
product_title	String	The title of the Product.
product_version	String	The version number of the Product.
query_string	String	The URL query string value on the inbound request.
rate_limit.count	Number	The number of API calls remaining in the specified rate limit time window.
rate_limit.limit	Number	The maximum number of requests an application is allowed to make to the API during a specified time window.

Field name	Type	Description
rate_limit.period	Number	The time window that is used to set a rate limit for API calls.
rate_limit.reject	String	An indication of whether calls that exceed the specified rate limit are rejected. If true, the API call is rejected with a 429 status code. If false, a record is created in the Activity log.
rate_limit.shared	String	An indication of whether the rate limit is shared at a Plan level by all operations, or whether a rate limit is specified on indivIDual operations.
request_body	String	The body of the inbound request.
request_http_headers.*field_name*	String	A component of the HTTP header section of the inbound request; for example, the acceptable encodings, the identification string for the user agent, or the proxies through which the request was sent. Note: The following types of headers are considered sensitive and do not show in analytics data for security reasons: • Any secret key configured in the API security • Any header that contains `secret` • Any header that contains `Authorization`
request_method	String	The method of the inbound request.
request_protocol	String	The protocol of the inbound request.
resource	String	The name of the operation.
resource_id	String	The operation identifier.
resource_path	String	The operation path.
response_body	String	The body of the outbound response.
response_http_headers.*field_name*	String	A component of the HTTP header section of the outbound response; for example, the MIME type of the content or the data and time when the message was sent.
status_code	String	The status code set on the outbound response.
time_to_serve_request	Number	The time elapsed (in milliseconds) from when the backend application received the request to when it sent a response.
transaction_id	String	The identifier for the API transaction. See https://www.ibm.com/docs/en/datapower-gateway/latest?topic=variables-varservicetransaction-id-servicevarstransactionid.
uri_path	String	The URI path on the inbound request.

Example API event record with payload logging set


```
{
  "@timestamp": "2024-03-11T12:30:09.105518124Z",
  "@version": "1",
  "api_id": "5973f48d-ee38-406f-bdea-5f04fa4b8dd8",
  "api_name": "p-2-p-bank-apis-api",
  "api_ref": "p-2-p-bank-apis-api:1.0.0",
  "api_version": "1.0.0",
    "app_id": "fd58ef46-de01-4d97-84b3-a2a536270224",
  "app_name": "API Connect App 4",
  "app_type": "production",
  "billing": {
    "amount": 0,
    "currency": "USD",
    "model": "free",
    "provider": "none",
    "trial_period_days": 0
  },
  "bytes_received": "4921",
  "bytes_sent": "4762",
  "catalog_id": "7ae861d0-f138-494e-898b-302ee981a03d",
  "catalog_name": "api-connect-catalog-1",
  "client_geoip": {
        "continent_code": "AS",
    "country_code2": "CN",
    "country_code3": "CN",
    "country_name": "China",
    "ip": "106.27.171.34",
    "latitude": 28.1783,
    "location": {
      "lat": 28.1783,
      "lon": 113.1117
    },
    "longitude": 113.1117,
    "region_code": "HN",
    "region_name": "Hunan",
    "timezone": "Asia/Shanghai"
  },
  "client_id": "8f7e3acb3e1da7d97e243635e03ede6a",
  "client_ip": "106.27.171.34",
  "datetime": "2024-03-11T11:12:36.387Z",
  "debug": [],
  "developer_org_id": "1ab7c606-23ff-40ec-9613-17804ef84c68",
  "developer_org_name": "api-connect-schowalter-abernathy-and-wehner-handmade-practical-cotton-chips-consumer-organization",
  "developer_org_title": "API Connect Schowalter, Abernathy and Wehner - Handmade Practical Cotton Chips Consumer
Organization",
  "endpoint_url": "N/A",
  "event_id": "0cf32181471ac5078fef486d21745278df2de4a1",
  "gateway_geoip": {
    "ip": "9.79.12.126",
    "country_code2": "US",
    "country_code3": "USA",
```

```
    "country_name": "United States",
    "continent_code": "NA",
    "city_name": "Durham",
    "postal_code": "27709",
    "latitude": 35.994,
    "longitude": -78.8986,
    "dma_code": 560,
    "area_code": 919,
    "timezone": "America/New_York",
        "region_name": "North Carolina",
    "location": {
      "lat": 35.994,
      "long": -78.8986
    }
  },
  "gateway_ip": "192.168.141.185",
  "host": "192.168.46.74",
  "http_user_agent": "Mozilla/5.0 (Linux; Android 8.1.0; SAMSUNG SM-G610M) AppleWebKit/537.36 (KHTML, like Gecko)
SamsungBrowser/23.0 Chrome/115.0.0.0 Mobile Safari/537.36",
  "immediate_client_ip": "8.46.111.156",
  "latency_info": [
    {
      "started": 0,
      "task": "Start"
    },
    {
      "started": 18,
      "task": "security-appID"
    },
    {
      "started": 19,
      "task": "PlanRateLimits"
    },
    {
      "started": 20,
      "task": "invoke"
    }
  ],
  "log_policy": "payload",
  "offload_max_queue": -1,
  "offload_queue_size": -1,
  "org_id": "c883b05a-82ca-45f2-a9d9-4696a22d9908",
  "org_name": "boborg",
  "plan_id": "api-connect-kunde-and-sons-sleek-intelligent-granite-fish-product:1.0.0:default-plan",
  "plan_name": "default-plan",
  "plan_version": "1.0.0",
  "product_id": "8df85619-2a6f-472e-a094-7abe7b059e83",
  "product_name": "api-connect-kunde-and-sons-sleek-intelligent-granite-fish-product",
  "product_ref": "api-connect-kunde-and-sons-sleek-intelligent-granite-fish-product:1.0.0",
  "product_title": "API Connect Kunde and Sons - Sleek Intelligent Granite Fish Product",
  "product_version": "1.0.0",
  "query_string": "",
  "rate_limit": {
    "rate-limit": {
      "interval": 1,
      "count": 1621,
      "reject": "true",
      "unit": "hour",
      "period": 3600,
      "shared": "true",
      "limit": 100
    }
  },
  "request_body": "",
  "request_http_headers": [
    {
      "Accept": "application/json, text/plain, */*"
    },
    {
      "Accept-Encoding": "gzip, compress, deflate, br"
    },
    {
      "Host": "cdm0310-gw.server.dev.example.com"
    },
    {
      "User-Agent": "Mozilla/5.0 (Linux; Android 8.1.0; SAMSUNG SM-G610M) AppleWebKit/537.36 (KHTML, like Gecko)
SamsungBrowser/23.0 Chrome/115.0.0.0 Mobile Safari/537.36"
    },
    {
      "Via": "1.1 BwAAAF4Hgnk-"
    },
    {
      "X-Client-IP": "9.46.111.156"
    },
    {
      "X-Forwarded-For": "106.27.171.34"
    },
    {
      "X-Global-Transaction-ID": "eba30d3065eef8d9000e5021"
    }
  ],
  "request_method": "GET",
  "request_protocol": "https",
  "resource_id": "p-2-p-bank-apis-api:1.0.0:get:/stores",
  "resource_path": "get",
  "response_body": "{ \"httpCode\":\"403\", \"httpMessage\":\"Forbidden\", \"moreInformation\":\"\" }",
```

```
  "response_http_headers": [
    {
      "Content-Type": "application/json"
    }
  ],
  "space_id": "d7442797-de03-4305-8f7e-6aa523589d1f",
  "space_name": "api-connect-space-2",
  "status_code": "403 Forbidden",
  "storage_max_queue": 7730941132,
  "storage_queue_size": 5162595,
  "tags": [
    "apicapievent",
    "_geoip_lookup_failure"
  ],
  "time_to_serve_request": "828",
  "transaction_id": "938017",
  "uri_path": "/boborg/api-connect-catalog-1/p-2-p-bank-apis-api/stores",
  "user_agent": {
    "device": "Samsung SM-G610M",
    "major": "23",
    "minor": "0",
    "name": "Samsung Internet",
    "os": "Android",
    "os_full": "Android 8.1.0",
    "os_major": "8",
    "os_minor": "1",
    "os_name": "Android",
    "os_patch": "0",
    "os_version": "8.1.0",
    "version": "23.0"
  },
  "gateway_geoip": {
    "ip": "9.79.12.126",
    "country_code2": "US",
    "country_code3": "USA",
    "country_name": "United States",
    "continent_code": "NA",
    "region_name": "NC",
    "city_name": "Durham",
    "postal_code": "27709",
    "latitude": 35.994,
    "longitude": -78.8986,
    "dma_code": 560,
    "area_code": 919,
    "timezone": "America/New_York",
    "real_region_name": "North Carolina",
    "location": [
      -78.8986,
      35.994
    ]
  }
}
```


Advanced configuration options

Configure advanced features and settings such as offload, retention period, and dedicated storage.

- **Dedicated storage and scaling up**
  Use dedicated storage for improved performance, stability, and scalability. Available on Kubernetes and OpenShift. Not available on VMware.
- **Retention and rollover**
  Manage how long your analytics subsystem stores data.
- **Logstash persistent queue**
  Enable the persistent queue feature to provide enhanced reliability in your API event data pipeline.
- **Offload to third-party systems**
  Offload your API event data for storage on a third-party system.
- **Disable local storage**
  If you are offloading you API event data to a third-party system, you can disable local storage to reduce subsystem resource requirements.
- **Shutting down the analytics subsystem**
  Disable the analytics service and shutdown the analytics subsystem.
- **Restarting analytics pods**
  How to restart the analytics pods in your API Connect deployment.
- **Change deployment profile**
  Change your deployment profile to switch between one replica and three replica deployments, or to change the computing resources available for your subsystem.
- **Performance tuning with ingestion pipeline properties**
  Fine tune your ingestion pipeline settings for optimal performance.

Dedicated storage and scaling up

Use dedicated storage for improved performance, stability, and scalability. Available on Kubernetes and OpenShift. Not available on VMware.

Note: Storage type selection is available on three replica deployments only.
To make an informed choice on which storage type is best for you, first read and understand the OpenSearch concepts that are described in OpenSearch nodes, indices, shards, and replicas.

API Connect provides two OpenSearch storage types:

- **Shared storage**
  A single pod provides data storage and OpenSearch cluster management. Shared storage is the default option, and the only option on a one replica deployment.
- **Dedicated storage**
  OpenSearch data storage and cluster management functions are split, and run in separate pods on each worker node:
  - The `storage-os-master` pod runs an OpenSearch "manager eligible" node that manages the OpenSearch cluster, but does not store any API event data.
  - The `storage` pod runs an OpenSearch "data" node that stores API event data.

Dedicated storage facilitates horizontal scaling to support greater analytics data storage and throughput. Add worker nodes and increase the number of `storage` pod replicas to scale horizontally (you do not need to increase the number of `storage-os-master` replicas). Dedicated storage also provides greater stability and allows some OpenSearch configuration changes to be made without downtime.
Note: For OpenShift users: The example steps in this topic use the Kubernetes `kubectl` command. On OpenShift, use the equivalent `oc` command in its place. If you are using a top-level CR you must edit the `APIConnectCluster` CR (the top-level CR), instead of directly in the subsystem CRs. If the subsystem section is not included in the top-level CR, copy and paste the section from the subsystem CR to the `APIConnectCluster` CR.
The storage type is defined in the analytics CR, in the `spec.storage.type` section as follows:

- Shared storage:

```
  storage:
    type: shared
    shared:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
        volumeSize: $DATA_VOLUME_SIZE
```

- Dedicated storage:

```
  storage:
    type: dedicated
    shared:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
        volumeSize: $DATA_VOLUME_SIZE
    master:
      volumeClaimTemplate:
        storageClassName: $STORAGE_CLASS
```

Changing from shared to dedicated storage

1. Edit your analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

2. Change storage type to dedicated:

   ```
     storage:
       type: dedicated
   ```

3. If your environment does not have a default storage class, or want to use a different storage class for your `storage-os-master` pods, then also add the `spec.storage.master` section to specify the storage class, for example:

   ```
     storage:
       type: dedicated
   ...
       master:
         volumeClaimTemplate:
           storageClassName: local_storage
   ```

Changing from dedicated to shared storage

1. Edit your analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

2. Change storage type to shared:

   ```
     storage:
       type: shared
   ```

3. Wait until all of your `storage` pods restart and are in the `Ready` state.
4. Edit your analytics CR and remove the `spec.storage.master` section.
5. Delete the `data-analytics-storage-os-master` PVCs:

   ```
   kubectl -n <namespace> delete pvc data-analytics-storage-os-master-<integer>
   ```

Horizontal scaling with dedicated storage

To take advantage of the horizontal scaling options with dedicated storage, see Horizontal scaling of storage pods.

Horizontal scaling of storage pods

Scale your storage pod replicas for increased API event data storage and throughput.

To take advantage of storage pod scaling, you must have a three replica deployment, and be using dedicated storage, see Dedicated storage and scaling up.

Note:

- The term "replica" when used with "data storage pod" refers to the Kubernetes storage pod, not OpenSearch replicas. OpenSearch replica settings are not configurable in API Connect.
- The OpenSearch term "node" refers to OpenSearch data storage nodes. OpenSearch nodes have a one-to-one relationship with the API Connect storage pod replicas, and have the same name.

This operation requires the use of the Analytics CLI.

Increase number of storage pod replicas

1. Edit your analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

2. Add the following to the `spec:` section:

   ```
     template:
     - name: storage
       replicaCount: <new replica count>
   ```

   where *<new replica count>* must be less than or equal to the number of worker nodes you have available.
3. Wait for the new storage pod replicas to start on your new worker nodes, and for the analytics cluster to rebalance your API event data to use your new replicas. The time the rebalancing takes depends on the size of your API event data and the network transmission speed between worker nodes. You can check the rebalancing status with the CLI health command:

   ```
   apic --mode analytics clustermgmt:getHealth  --server <platform api endpoint> --analytics-service <analytics service name>
   --format json
   ```

   where:
   - `<platform api endpoint>` is the platform API endpoint.
   - `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

   When rebalancing is complete, the `number_of_data_nodes` matches your updated `replicaCount`, and the highlighted `_shards` properties are all zero, as shown:

   ```
   {
      "cluster_name":"apic-analytics-cluster",
      "status":"green",
      "timed_out":false,
      "number_of_nodes":8,
      "number_of_data_nodes":5,
      "discovered_master":true,
      "discovered_cluster_manager":true,
      "active_primary_shards":8,
      "active_shards":24,
      "relocating_shards":0,
      "initializing_shards":0,
      "unassigned_shards":0,
      "delayed_unassigned_shards":0,
      "number_of_pending_tasks":0,
      "number_of_in_flight_fetch":0,
      "task_max_waiting_in_queue_millis":0,
      "active_shards_percent_as_number":100
   }
   ```

Note: If in the past you increased and then reduced the number of data storage replicas in your deployment, you might have an OpenSearch exclude list that contains the names of the previous replicas you removed. New storage pod replicas cannot be created if their name is in the OpenSearch exclude list. To empty the OpenSearch exclude list, follow the same steps that you used to create it: Create exclude list, but set `persistent.cluster.routing.allocation.exclude._name` to an empty string.

Reduce number of storage pod replicas

1. Verify that your analytics cluster health is green. Use the analytics CLI health command:

   ```
   apic --mode analytics clustermgmt:getHealth  --server <platform api endpoint> --analytics-service <analytics service name>
   --format json
   ```

   where:
   - `<platform api endpoint>` is the platform API endpoint.
   - `<analytics service name>` is the name of the analytics service as configured in the Cloud Manager UI.

   Check that the status property is green:

   ```
   {
      "cluster_name":"apic-analytics-cluster",
      "status":"green",
      "timed_out":false,
      "number_of_nodes":8,
      "number_of_data_nodes":5,
      "discovered_master":true,
      "discovered_cluster_manager":true,
      "active_primary_shards":8,
      "active_shards":24,
      "relocating_shards":0,
      "initializing_shards":0,
      "unassigned_shards":0,
      "delayed_unassigned_shards":0,
      "number_of_pending_tasks":0,
   ```

```
   "number_of_in_flight_fetch":0,
   "task_max_waiting_in_queue_millis":0,
   "active_shards_percent_as_number":100
}
```

Do not proceed if the status is not green.

2. Replicas are removed by reducing the Kubernetes **replicaCount** in the analytics CR. When **replicaCount** is reduced, the replicas that are deleted are always the highest numbered replicas. For example:

```
kubectl get pods | grep storage
```

```
NAME                                  READY   STATUS      RESTARTS   AGE
analytics-storage-0                   1/1     Running     0          140d
analytics-storage-1                   1/1     Running     0          140d
analytics-storage-2                   1/1     Running     0          140d
analytics-storage-3                   1/1     Running     0          140d
analytics-storage-4                   1/1     Running     0          140d
```

If you reduce **replicaCount** from 5 to 3, the two replicas that are deleted are **analytics-storage-3** and **analytics-storage-4**.
Before you reduce the **replicaCount**, you must ensure that replicas to be deleted contain no API event data. To make OpenSearch relocate all API event data to the replicas you are keeping, add the replicas that you want to delete to the OpenSearch exclude list:

a. Export the current analytics cluster settings to a JSON file:

```
apic -m analytics clustermgmt:getSettings --server <platform api endpoint> --analytics-service <analytics service
name> --format json > a7s_cluster_settings.json
```

This command creates a file called a7s_cluster_settings.json in your current directory.

b. Edit the a7s_cluster_settings.json file, and set the **persistent.cluster.routing.allocation.exclude._name** property to a comma-separated list of replicas you plan to remove. For example:

```
{
    "persistent": {
        "cluster": {
            "routing": {
                "allocation": {
                    "exclude": {
                        "_name": "analytics-storage-3,analytics-storage-4"
                    }
                }
            }
        },
        "plugins": {
            "index_state_management": {
                "metadata_migration": {
                    "status": "1"
                },
                "template_migration": {
                    "control": "-1"
                },
                "history": {
                    "enabled": "false"
                }
            }
        }
    },
    "transient": {}
}
```

You might need to add the **cluster.routing.allocation.exclude** structure if this property has not been set before.

c. Apply the updated a7s_cluster_settings.json file:

```
apic -m analytics clustermgmt:putSettings --server <platform api endpoint> --analytics-service <analytics service
name> a7s_cluster_settings.json --format json
```

3. Wait for your analytics cluster to remove all data from the replicas you want to delete, which you specified in step 2. You can check that the data rebalancing is complete with the **clustermgmt:catAllocation** CLI command:

```
apic -m analytics clustermgmt:catAllocation --server <platform api endpoint> --analytics-service <analytics service name>
--format json --return_format json
```

Replicas that don't have data appear as entries with a **shards** value of zero in the output (you can ignore the **disk.*** properties):

```
[
  {
    "shards": "5",
    "disk.indices": "113.3kb",
    "disk.used": "27.3gb",
    "disk.avail": "222.5gb",
    "disk.total": "249.8gb",
    "disk.percent": "10",
    "host": "192.168.87.226",
    "ip": "192.168.87.226",
    "node": "analytics-storage-0"
  },
  {
    "shards": "0",
    "disk.indices": "0",
    "disk.used": "36.9kb",
    "disk.avail": "212.9kb",
    "disk.total": "249.8kb",
    "disk.percent": "1",
    "host": "192.168.64.225",
```

```
    "ip": "192.168.64.225",
    "node": "analytics-storage-3"
  },
  {
    "shards": "5",
    "disk.indices": "16.8kb",
    "disk.used": "28.2gb",
    "disk.avail": "221.6gb",
    "disk.total": "249.8gb",
    "disk.percent": "11",
    "host": "192.168.112.227",
    "ip": "192.168.112.227",
    "node": "analytics-storage-1"
  },
  {
    "shards": "5",
    "disk.indices": "23.3kb",
    "disk.used": "29.3gb",
    "disk.avail": "220.4gb",
    "disk.total": "249.8gb",
    "disk.percent": "11",
    "host": "192.168.24.102",
    "ip": "192.168.24.102",
    "node": "analytics-storage-2"
  },
  {
    "shards": "0",
    "disk.indices": "0",
    "disk.used": "35.4kb",
    "disk.avail": "214.4kb",
    "disk.total": "249.8kb",
    "disk.percent": "1",
    "host": "192.168.91.149",
    "ip": "192.168.91.149",
    "node": "analytics-storage-4"
  }
]
```

The time rebalancing takes depends on the size of your API event data and the network transmission speed between pod replicas.
Note: The value in the **node** property of the **clustermgmt:catAllocation** output is the same as the names of the Kubernetes storage pod replicas.

4. Verify that your analytics cluster health is green. Use the analytics CLI health command:

```
apic --mode analytics clustermgmt:getHealth  --server <platform api endpoint> --analytics-service <analytics service name>
--format json
```

where:

- ***<platform api endpoint>*** is the platform API endpoint.
- ***<analytics service name>*** is the name of the analytics service as configured in the Cloud Manager UI.

Check that the status property is green:

```
{
   "cluster_name":"apic-analytics-cluster",
   "status":"green",
   "timed_out":false,
   "number_of_nodes":8,
   "number_of_data_nodes":5,
   "discovered_master":true,
   "discovered_cluster_manager":true,
   "active_primary_shards":8,
   "active_shards":24,
   "relocating_shards":0,
   "initializing_shards":0,
   "unassigned_shards":0,
   "delayed_unassigned_shards":0,
   "number_of_pending_tasks":0,
   "number_of_in_flight_fetch":0,
   "task_max_waiting_in_queue_millis":0,
   "active_shards_percent_as_number":100
}
```

Do not proceed if the status is not green.

5. Edit your analytics CR:

```
kubectl -n <namespace> edit a7s
```

6. Update the **replicaCount**:

```
  template:
  - name: storage
    replicaCount: <new replica count>
```

where *<new replica count>* is the reduced number of replicas that you want.
Note: If the **replicaCount** property does not exist in the analytics CR, check that you are not attempting to reduce **replicaCount** to 2. Reducing **replicaCount** to less than 3 is not a supported operation.

7. Monitor the health status of your analytics cluster with the CLI health command:

```
apic --mode analytics clustermgmt:getHealth  --server <platform api endpoint> --analytics-service <analytics service name>
--format json
```

where:

- ***<platform api endpoint>*** is the platform API endpoint.
- ***<analytics service name>*** is the name of the analytics service as configured in the Cloud Manager UI.

Confirm that the **status** is **green**, the **number_of_data_nodes** matches your updated **replicaCount**, and that the highlighted shard properties are all zero:

```
{
   "cluster_name":"apic-analytics-cluster",
   "status":"green",
   "timed_out":false,
   "number_of_nodes":6,
   "number_of_data_nodes":3,
   "discovered_master":true,
   "discovered_cluster_manager":true,
   "active_primary_shards":8,
   "active_shards":24,
   "relocating_shards":0,
   "initializing_shards":0,
   "unassigned_shards":0,
   "delayed_unassigned_shards":0,
   "number_of_pending_tasks":0,
   "number_of_in_flight_fetch":0,
   "task_max_waiting_in_queue_millis":0,
   "active_shards_percent_as_number":100
}
```

Retention and rollover

Manage how long your analytics subsystem stores data.

Each API event is recorded as a document in an OpenSearch index. When this index reaches a maximum size or age it is archived, and a new index is created for recording new API events. This is known as index rollover. When an archived index reaches a maximum age, which is known as the data retention maximum, it is deleted. You can specify the index rollover size and the data retention period in the Cloud Manager UI.

To learn more about OpenSearch indices, see OpenSearch nodes, indices, shards, and replicas.

The index rollover settings determine how long an index accumulates data before it "rolls over" and a new index is created. You can control the duration of an index two ways: by setting a maximum age for the index (default is 1 day), or by setting a maximum number of documents that can be recorded in the index (default is 25 million documents). When the rollover occurs, a new index is created for new API events to be stored in, while the previous index is archived and kept until its age exceeds the data retention setting. If you change the rollover settings, consider both the amount of data to be stored within each index and the total number of indices. Allowing indices to grow too large, or storing too many indices, can cause performance problems.

The data retention setting determines how long the stored indices are retained. When an index's creation date exceeds the retention period, that index and all its data is deleted.
Note: It is the creation date of the index that is used to determine retention, not the age of the newest record in the index.
The default retention period is 30 days for a one replica deployment, and 90 days for a three replica deployment. Reasons for changing this setting might include storage constraints, and the data retention requirements for your organization. You might want to set this value to be less than the default if you have a high transaction load and limited persistent storage space, especially if payload logging is enabled on many of your APIs.

The most recently created index (the index that is currently being written to) is not deleted, even if you set the retention period as small as 1 hour. If you regularly need to delete data quickly, adjust the index rollover so that a new index is rolled over to sooner, and the old index can be deleted.

Before an index is deleted, the API event data in the index is summarized and stored in a separate summary index to provide long-term data. The summary index has a separate retention setting, which is set to one year by default.

Note: The rollover and retention settings have a significant impact on the memory that is required by the analytics subsystem. If you have rollover set to one day (1d), then to maintain an acceptable level of performance the following retention settings are recommended for each deployment profile:

- **`n1xc2.m16`**: This profile provides 3 GB of heap memory for storage on one node. Recommended retention is four days or less.
- **`n1xc4.m32`**: This profile provides 11 GB of heap memory for storage on one node. Recommended retention is 15 days or less.
- **`n1xc6.m48`**: This profile provides 19 GB of heap memory for storage on one node. Recommended retention is 25 days or less.
- **`n3xc4.m16`**: This profile provides 3 GB of heap memory for storage on three nodes. Recommended retention is 12 days or less.
- **`n3xc4.m32`**: This profile provides 11 GB of heap memory for storage on three nodes. Recommended retention is 45 days or less.
- **`n3xc6.m48`**: This profile provides 19 GB of heap memory for storage on three nodes. Recommended retention is 76 days or less.
- **`n3xc8.m64`**: This profile provides 27 GB of heap memory for storage on three nodes. Recommended retention is 108 days or less.

If rollover is set to higher than one day, then more analytics data can be stored before a performance deterioration is observed. However, a higher rollover setting has the disadvantage that more data is deleted when an index reaches the retention age. For example, if you set your rollover period to seven days and you have the default 90-day retention period, then when your oldest index reaches the 90 days since it was created, it is deleted and you are left with 83 days of API event data.
To view and update the rollover and retention settings, use the Cloud Manager UI.

1. From the Cloud Manager UI, click Topology.
2. Select the Analytics Service you want to view or edit.
3. Click Advanced settings.

Alternative method using the Analytics CLI.

1. View the current retention and rollover settings by using the **`clustermgmt:getStorageSettings`** analytics CLI command:

   ```
   apic -m analytics clustermgmt:getStorageSettings --server <platform api endpoint> --analytics-service <analytics service name> --format json
   ```

   - ***`<platform api endpoint>`*** is the platform API endpoint.
   - ***`<analytics service name>`*** is the name of analytics service as configured in the Cloud Manager UI.
2. To update the retention and rollover settings create a JSON file that is called **`retention_and_rollover_settings.json`**, paste in the output from step 1 and set the retention and rollover setting as required. Apply these new settings with: **`clustermgmt:putStorageSettings`**

   ```
   apic -m analytics clustermgmt:putStorageSettings --server <platform api endpoint> --analytics-service <analytics service name> retention_and_rollover_settings.json
   ```

- `<platform api endpoint>` is the platform API endpoint.
- `<analytics service name>` is the name of analytics service as configured in the Cloud Manager UI.

Logstash persistent queue

Enable the persistent queue feature to provide enhanced reliability in your API event data pipeline.

Note: From v10.0.7 the persistent queue feature is enabled by default.
If you want to disable this feature, set the `ingestion.queue.type` property to "memory". If you plan to re-enable this feature in the future, then leave the `volumeClaimTemplate` section as it is. If you do not expect to enable this feature again, then delete the `volumeClaimTemplate` section, and the associated PVC.

The Logstash persistent queue provides enhanced reliability in the data pipeline. The ingestion pod buffers incoming API event data to persistent storage so that no events are lost if the downstream local storage pods or offload targets are temporarily unavailable.

For example, when you restart the analytics storage pods they take a few seconds to be able to receive API event data. While the storage pods are restarting, incoming API events are held in the persistent queue until the storage pods are available again. If the storage pods are unavailable and persistent queue is not enabled, then new API event data is lost during the storage pod outage.

Another benefit of the persistent queue feature is better handling of unexpected traffic spikes. The only disadvantage of the persistent queue feature is greater computing resource requirements for the ingestion pods.

Enable persistent queue on Kubernetes, OpenShift and Cloud Pak for Integration

1. **Kubernetes and OpenShift individual subsystem installations:**
   a. Edit your analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

   b. Add the persistent queue settings to the `spec.ingestion` section:

   ```
     queue:
       type: persisted
       volumeClaimTemplate:
         storageClassName: <storage-class-name>
         volumeSize: 50Gi # Do not change this value
   ```

   where *<storage-class-name>* is the Kubernetes storage class that is defined for persistent volumes. For example, `local-storage`.
2. **Cloud Pak for Integration and OpenShift top-level CR installations:**
   a. Get the contents of your analytics CR:

   ```
   kubectl -n <namespace> get a7s -o yaml
   ```

   b. Copy the `spec.ingestion` section to a text editor, and indent each line by two extra spaces. For example:

   ```
     ingestion:
       clientAuthEnabled: true
       clientSubjectDN: CN=a7s-ing-client
       configReloadAutomatic: true
       enabled: true
       endpoint:
         annotations:
           cert-manager.io/issuer: instname-ingress-issuer
         hosts:
         - name: instname-a7s-ai-endpoint-apic.apps.iaintlcr.cp.fyre.ibm.com
           secretName: instname-a7s-ai-endpoint
       pipelineBatchDelay: 10
       pipelineBatchSize: 2000
       pipelineWorkers: 4
   ```

   c. Add the persistent queue settings to the ingestion section in your text editor:

   ```
     ingestion:
       clientAuthEnabled: true
       clientSubjectDN: CN=a7s-ing-client
       configReloadAutomatic: true
       enabled: true
       endpoint:
         annotations:
           cert-manager.io/issuer: instname-ingress-issuer
         hosts:
         - name: instname-a7s-ai-endpoint-apic.apps.iaintlcr.cp.fyre.ibm.com
           secretName: instname-a7s-ai-endpoint
       pipelineBatchDelay: 10
       pipelineBatchSize: 2000
       pipelineWorkers: 4
       queue:
         type: persisted
         volumeClaimTemplate:
           storageClassName: <storage-class-name>
           volumeSize: 50Gi # Do not change this value
   ```

   where *<storage-class-name>* is the Kubernetes storage class that is defined for persistent volumes. For example, `local-storage`.
   d. Edit your `APIConnectCluster` CR:

   ```
   kubectl -n <namespace> edit APIConnectCluster
   ```

e. Copy the updated ingestion section from your text editor to your `APIConnectCluster` CR.

3. Save you changes to the CR. The ingestion pods restart automatically.

Enable persistent queue on OVA

Configure the persistent queue in the analytics-extra-values.yaml file.

1. From your project directory, open the analytics-extra-values.yaml file for editing. If this file does not exist, then create it: Analytics extra-values file.
2. Add the following text to the file, inside the `spec` object:

```
ingestion:
  queue:
     type: persisted
     volumeClaimTemplate:
        storageClassName: local-storage
        volumeSize: 50Gi
```

3. Ensure that your `extra-values-file` property is set to point to your analytics-extra-values.yaml file:

```
apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
```

4. Apply the analytics-extra-values.yaml to your analytics subsystem:

```
apicup subsys install <analytics subsystem>
```

Offload to third-party systems

Offload your API event data for storage on a third-party system.

The analytics event data that is sent from the gateway to analytics can be offloaded to a third-party system in real time. The third-party system can provide enhanced monitoring, alerting, storage, and analysis, in addition to what the built-in analytics subsystem provides. Multiple target systems are supported, and the data can be modified before offloading.

APIC supports offloading analytics data to the following third-party systems:

- HTTP endpoints
- OpenSearch
- Elasticsearch
- Apache Kafka
- Syslog servers

If you do not need internal storage and want to use only the third-party system for working with your analytics data, consider disabling internal storage entirely as explained in Disable local storage.

- **Configuring offload on Kubernetes and OpenShift**
  Configure your analytics subsystem to offload API event data to a third-party system.
- **Configuring offload on VMware**
  Configure your VMware analytics subsystem to offload API event data to a third-party system.
- **Sample output plugins for offloading data**
  Review sample output plugins to see how to code your own plugins for offloading data from API Connect analytics.
- **Using offloaded analytics data to train AI models**
  How to use API Connect analytics data to train AI models.
- **Troubleshooting offload**
  Checks to help determine the cause of offload failure.

Configuring offload on Kubernetes and OpenShift

Configure your analytics subsystem to offload API event data to a third-party system.

Procedure

1. Edit your analytics CR:

```
kubectl edit a7s
```

Note: If you are using the OpenShift top-level CR deployment, edit your `APIConnectCluster` CR instead. Updates should be made in the `spec.analytics` section.

2. Set `spec.external.offload` to true, and add the offload plugin configuration:
The following example shows an HTTP output plugin:

```
  external:
    offload:
      enabled: true
      output: |
        http {
          url => "https://example.com"
          http_method => "post"
          ssl_verification_mode => "none"
          id => "API Connect analytics event data"
        }
```

In this example, the URL points to an HTTPS endpoint, but verification of the endpoint's server certificate is disabled by setting `ssl_verification_mode` to "none". If you want your analytics subsystem to verify the endpoint server certificate, then see Configuring offload server certificates on Kubernetes and OpenShift.

The `id` property is not mandatory, but it is recommended.

For more information about the HTTP output plugin configuration, see the Logstash HTTP plugin documentation.

3. Optional: Add a `filter` after the output plugin configuration to modify data before it is sent to the offload target.
The following example filter removes the `query_string` field:

```
external:
  offload:
    enabled: true
    output: |
      http {
        ...
      }
    filter: |
      ruby {
        remove_field => ["query_string"]
      }
```

For information about modifying offloaded data, see Customizing data with ingestion filters.
4. Save the changes to your analytics CR. The ingestion pods restart automatically.
5. Update your analytics subsystem infrastructure backup with your updated analytics CR. See Backup, restore, and disaster recovery.

- **Configuring offload server certificates on Kubernetes and OpenShift**
Configure your analytics subsystem to trust your offload endpoint.
- **Configuring offload client certificates on Kubernetes and OpenShift**
Configuring client keys for analytics offload.

Configuring offload server certificates on Kubernetes and OpenShift

Configure your analytics subsystem to trust your offload endpoint.

Before you begin

You must have the offload target's server certificate in either of these formats:

- Base64 encoded `pem` file.
- JKS file. If you are using a JKS file, encode the contents in Base64:

```
cat <JKS certificate filename> | base64
```

About this task

The ingestion pod of your analytics subsystem sends your API event data to your offload target. If your offload target uses TLS, then your ingestion pod must be configured to trust the offload target. The steps in this procedure insert the server certificate in the ingestion pod's offload certificates directory, so that the ingestion pod can validate the certificate it receives from the offload endpoint.

Procedure

1. Create a file called analytics_offload_certificates.yaml, and paste in the following contents:
   - If you are using a JKS file, then in the `data` section of the secret, paste the encoded JKS string as shown:

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-certificates
data:
  keystore.jks: "<base64 encoded JKS string>"
```

   - If you are using a PEM file, then in the `stringData.cacert` section paste the contents of the PEM file.

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-certificates
stringData:
  cacert: |-
    -----BEGIN CERTIFICATE-----
    MIIDIDCCAomgAwIBAgIENd70zzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJVUzEQMA4GA1UE
    ChMHRXF1aWZheDEtMCsGA1UECxMkRXF1aWZheCBTZWN1cmUgQ2VydGlmaWNhdGUgQXV0aG9yaXR5
    MB4XDTk4MDgyMjE2NDE1MVoXDTE4MDgyMjE2NDE1MVowTjELMAkGA1UEBhMCVVMxEDAOBgNVBAoT
    B0VxdWlmYXgxLTArBgNVBAsTJEVxdWlmYXggU2VjdXJlIENlcnRpZmljYXRlIEF1dGhvcml0eTCB
    nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwV2xWGcIYu6gmi0fCG2RFGiYCh7+2gRvE4RiIcPR
    fM6fBeC4AfBONOziipUEZKzxa1NfBbPLZ4C/QgKO/t0BCezhABRP/PvwDN1Dulsr4R+AcJkVV5MW
    8Q+XarfCaCMczE1ZMKxRHjuvK9buY0V7xdlfUNLjUA86iOe/FP3gx7kCAwEAAaOCAQkwggEFMHAG
    A1UdHwRpMGcwZaBjoGGkXzBdMQswCQYDVQQGEwJVUzEQMA4GA1UEChMHRXF1aWZheDEtMCsGA1UE
    CxMkRXF1aWZheCBTZWN1cmUgQ2VydGlmaWNhdGUgQXV0aG9yaXR5MQ0wCwYDVQQDEwRDUkwxMBoG
    A1UdEAQTMBGBDzIwMTgwODIyMTY0MTUxWjALBgNVHQ8EBAMCAQYwHwYDVR0jBBgwFoAUSOZo+SvS
    spXXR9gjIBBPM5iQn9QwHQYDVR0OBBYEFEjmaPkr0rKV10fYIyAQTzOYkJ/UMAwGA1UdEwQFMAMB
    Af8wGgYJKoZIhvZ9B0EABA0wCxsFVjMuMGMDAgbAMA0GCSqGSIb3DQEBBQUAA4GBAFjOKer89961
    zgK5F7WF0bnj4JXMJTENAKaSbn+2kmOeUJXRmm/kEd5jhW6Y7qj/WsjTVbJmcVfewCHrPSqnI0kB
```

```
    BIZCe/zuf6IWUrVnZ9NA2zsmWLIodz2uFHdh1voqZiegDfqnc1zqcPGUIWVEX/r87yloqaKHee95
    70+sB3c4
    -----END CERTIFICATE-----
```

2. Create the `offload-secret` in your analytics namespace, from the analytics_offload_certificates.yaml file.

   ```
   kubectl -n <namespace> apply -f analytics_offload_certificates.yaml
   ```

   Verify that the `offload-secret` was created:

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

3. Edit your analytics CR:

   ```
   kubectl edit a7s
   ```

   Note: If you are using the OpenShift top-level CR deployment, edit your `APIConnectCluster` CR instead. Updates should be made in the `spec.analytics` section.
4. Add the new property `spec.external.offload.secretName`, and set it to `offload-secret` (the secret you created in step 2).

   ```
   offload:
     enabled: true
     secretName: offload-secret
     output: |
       ...
   ```

   When your ingestion pod restarts, it copies the contents of the offload-secret into its /etc/velox/external_certs/offload directory.

5. Update your plugin properties to include the path to the server certificate. The property name for the endpoint server certificate depends on your output target. For HTTP the property name is `ssl_certificate_authorities`. Consult the Logstash documentation to confirm the property name used by other plugin types such as Syslog and Elasticsearch.
   Example HTTP output plugin:

   ```
   ...
         output: |
           http {
             url => "https://example.com"
             http_method => "post"
             id => "API Connect analytics event data"
             ssl_certificate_authorities => "/etc/velox/external_certs/offload/<certname>"
           }
   ```

   where *<certname>* is the name that you specified in the analytics_offload_certificates.yaml file in step 1, either `cacert` or `keystore.jks` depending on the format of certificate used.
6. Save the updated analytics CR.
   When you save the updated CR, your ingestion pod restarts itself automatically and detects the new `secretName` in the CR. On startup, the ingestion pod copies the certificate from the specified `secretName` to its /etc/velox/external_certs/offload directory. You can verify that the certificate was successfully imported with:

   ```
   kubectl exec -ti <ingestion pod> -- bash -ic "ls /etc/velox/external_certs/offload"
   ```

   This command should return the certificate name that you specified in the analytics_offload_certificates.yaml file, either `cacert` or `keystore.jks` depending on the format of certificate used.
   When your ingestion pod communicates with your HTTPS endpoint, it verifies the server certificate that the endpoint returns against the certificate that is stored in /etc/velox/external_certs/offload.

7. Ensure that you take a backup of your offload secret and your updated analytics CR. For more information about backups, see Backup, restore, and disaster recovery

Configuring offload client certificates on Kubernetes and OpenShift

Configuring client keys for analytics offload.

The topic Configuring offload server certificates on Kubernetes and OpenShift covers how to configure your ingestion pod to trust your offload endpoint server certificate. This topic covers how to configure mTLS client certificates to be used when your ingestion pod communicates with the offload endpoint.

To configure client certificates, follow the same steps as Configuring offload server certificates on Kubernetes and OpenShift, but add your client certificate and key to the analytics_offload_certificates.yaml file.

For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: offload-certificates
stringData:
  cacert: |-
    -----BEGIN CERTIFICATE-----
    ...
  clientcert: |-
    -----BEGIN CERTIFICATE-----
    ...
  clientkey: |-
    -----BEGIN CERTIFICATE-----
    ...
```

All certificates added to the `offload-certificates` secret are created as individual files in the ingestion pod's /etc/velox/external_certs/offload directory. You can confirm that the certificate files were created with this command:

```
kubectl exec -ti <ingestion pod> -- bash -ic "ls /etc/velox/external_certs/offload"
```

In this example, the command output that is returned shows the three certificate files corresponding to the ones specified in analytics_offload_certificates.yaml:

```
cacert  clientcert  client
```

Update the offload plugin configuration in your analytics CR to reference the certificates in /etc/velox/external_certs/offload as required.

Example HTTP plugin that specifies client certificates:

```
  external:
    offload:
      enabled: true
      output: |
        http {
          url => "https://example.com"
          http_method => "post"
          ssl_certificate_authorities => "/etc/velox/external_certs/offload/cacert"
          ssl_certificate => "/etc/velox/external_certs/offload/clientcert"
          ssl_key => "/etc/velox/external_certs/offload/clientkey"
          id => "API Connect analytics event data"
        }
```

For other plugin types the TLS property names might differ, check the Logstash documentation.
Important: After updating your offload configuration, take a new analytics infrastructure configuration backup to ensure that you have backups of your analytics_offload_certificates.yaml file, your offload secret, and your analytics CR. See Backup, restore, and disaster recovery.

Configuring offload on VMware

Configure your VMware analytics subsystem to offload API event data to a third-party system.

Procedure

1. From your project directory, open the analytics-extra-values.yaml file for editing. If this file does not exist, then create it: Analytics extra-values file.
2. Add the following text to the file, inside the **spec** section:

   ```
   spec:
   ...
     external:
       offload:
         enabled: true
         output: |
           <output definition>
   ```

   where *<output definition>* is the configuration for the chosen output plugin that will send data to the target third-party system.The following example enables offloading to HTTP:

   ```
   spec:
     external:
       offload:
         enabled: true
         output: |
           http {
             url => "https://example.com"
             http_method => "post"
             ssl_verification_mode => "none"
             id => "API Connect analytics event data"
           }
   ```

   In this example, the URL points to an HTTPS endpoint, but verification of the endpoint's server certificate is disabled by setting **ssl_verification_mode** to "none". If you want your analytics subsystem to verify the endpoint server certificate, then see Configuring offload server certificates on VMware.

   The **id** property is not mandatory, but it is recommended.

   For more information about the HTTP output plugin configuration, see the Logstash HTTP plugin documentation.

3. Optional: Add a **filter** after the output plugin to modify data before it is sent to the offload target.
   The following example filter removes the **query_string** field:

   ```
   spec:
     external:
       offload:
         enabled: true
         output: |
           http {
             url => "example.com"
             http_method => "post"
             id => "offload_http"
           }
         filter: |
           ruby {
             remove_field => ["query_string"]
           }
   ```

   For information about modifying offloaded data, see Customizing data with ingestion filters.

4. Save the analytics-extra-values.yaml file.
5. Ensure that your `extra-values-file` property is set to point to your analytics-extra-values.yaml file:

   ```
   apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
   ```

6. Apply the analytics-extra-values.yaml to your analytics subsystem:

   ```
   apicup subsys install <analytics subsystem>
   ```

7. Take a backup of your updated analytics-extra-values.yaml file. For more information about backups, see Backup, restore, and disaster recovery.

- **Configuring offload server certificates on VMware**
  Configure your analytics subsystem to trust your offload endpoint.
- **Configuring offload client certificates on VMware**
  Configuring client keys for analytics offload.

Configuring offload server certificates on VMware

Configure your analytics subsystem to trust your offload endpoint.

Before you begin

You must have the offload target's server certificate in either of these formats:

- Base64 encoded `pem` file.
- JKS file. If you are using a JKS file, encode the contents in Base64:

  ```
  cat <JKS certificate filename> | base64
  ```

Procedure

1. SSH to your analytics VM, or one of the VMs if you have a three replica deployment.

   ```
   ssh apicadm@<analytics VM FQDN>
   ```

2. Switch to the root user:

   ```
   sudo -i
   ```

3. If you are using a JKS file, create a file to contain your certificate.

   ```
   echo "<certificate>" > <certificate filename>
   ```

4. Follow the steps for Configuring offload server certificates on Kubernetes and OpenShift.
   When you follow Kubernetes steps on VMware, run all operations as the root user. You do not need to specify a namespace when running **kubectl** commands on VMware. For example if a step has the command:

   ```
   kubectl -n <namespace> apply -f analytics_offload_certificates.yaml
   ```

   you can omit the `-n <namespace>`.
   To identify the ingestion pod, run:

   ```
   kubectl get pods
   ```

5. Keep a backup of the server certificate as part of your analytics infrastructure backup. For more information about backups, see Backup, restore, and disaster recovery.

Configuring offload client certificates on VMware

Configuring client keys for analytics offload.

Procedure

1. SSH to your analytics VM, or one of the VMs if you have a three replica deployment.

   ```
   ssh apicadm@<analytics VM FQDN>
   ```

2. Switch to the root user:

   ```
   sudo -i
   ```

3. Refer to the Kubernetes steps documented in Configuring offload client certificates on Kubernetes and OpenShift.
   When you follow Kubernetes steps on VMware, run all operations as the root user. You do not need to specify a namespace when running **kubectl** commands on VMware. For example if a step has the command:

   ```
   kubectl -n <namespace> apply -f analytics_offload_certificates.yaml
   ```

   you can omit the `-n <namespace>`.
   To identify the ingestion pod, run:

```
kubectl get pods
```

4. Keep a backup of your client certificates as part of your analytics infrastructure backup. For more information about backups, see Backup, restore, and disaster recovery.

Sample output plugins for offloading data

Review sample output plugins to see how to code your own plugins for offloading data from API Connect analytics.

For complete details of output plugin syntax, see the Logstash output plugins documentation.

HTTP

Example HTTP output plug-in:

```
http {
   url => "example.com"
   http_method => "post"
   codec => "json"
   content_type => "application/json"
   headers => {
      "x-acme-index" => "8"
      "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
   }
   id => "offload_http"
}
```

Example HTTP output plug-in that references a custom certificate for accessing the third-party endpoint:

```
http {
   url => "https://example.com:443"
   http_method => "post"
   codec => "json"
   content_type => "application/json"
   headers => {
      "x-acme-index" => "8"
      "x-acme-key" => "0d5d259f-b8c5-4398-9e58-77b05be67037"
   }
   id => "offload_http"
   ssl_certificate_authorities => "/etc/velox/external_certs/offload/cacert"
}
```

For more information about offloading to HTTP, refer to the HTTP output plugin documentation.

Elasticsearch

To configure data offload to an Elasticsearch cluster, you must specify the following properties:

- One or more server Elasticsearch server URLs.
- A dynamically configured name to define how indices are created.
- The number of primary shards for storing the indexed documents.
- The number of replica shards for redundancy.
- Server authentication credentials (if requried).

The naming syntax that you specify for your indices determines how many indices are created and how they are characterized. For example, you can define indices that are created based on a date pattern, or indices that are created to store data by provider organization or by API name.

Example Elasticsearch output plugin:

```
elasticsearch {
      hosts => "https://example.com:443"
      index => "apiconnect"
      ssl  => true
      cacert => "/etc/velox/external_certs/offload/cacert"
}
```

For more information about offloading to Elasticsearch, refer to the Elasticsearch output plugin documentation.

Apache Kafka

To configure data offload to a Kafka cluster, provide host and port connection details for one or more servers, and the name of the Kafka topic. To be able to identify API Connect data, include the string identifier `id`.

Example Kafka output plug-in:

```
kafka {
      topic_id => "api_events"
      bootstrap_servers => "example.com:9093"
      codec => "json"
      id => "apic_kafka_offload"
      ssl_truststore_location => "/usr/share/logstash/jdk/lib/security/cacerts"
      ssl_truststore_password => "changeit"
      ssl_truststore_type => "JKS"
      security_protocol => SSL
}
```

For more information on offloading to Apache Kafka, refer to the Kafka output plugin documentation.

Syslog

To configure data offload to Syslog, you must provide host and port connection details for the server.

Restriction: Syslog does not support the use of chained client certificates for TLS profiles.
Example Syslog output plug-in:

```
syslog {
  host => "example.com"
  port => 601
  protocol => "ssl-tcp"
  id => "offload_syslog"
  appname => "apiconnect"
  msgid => "%{org_id}"
  facility => "user-level"
  severity => "informational"
  codec => "json"
  ssl_cacert => "/etc/velox/external_certs/offload/cacert"
}
```

Attention: Due to a third-party issue with Logstash, most of the facility values are not set correctly on the offload server. As a temporary workaround, set the facility to "user-level" to ensure that the corresponding value is correctly sent to the offload server. When Logstash corrects this problem, you must update the offload configuration to change the value to a more appropriate setting.
For more information about offloading to Syslog, refer to the Syslog output plugin documentation.

OpenSearch

To configure data offload to OpenSearch, see the Ship to OpenSearch documentation.

Noname

To configure data offload to Noname, see the Noname documentation.

Example Noname offload plugin:

```
http {
  url => "https://<ENGINE_URL>/engine?structure=ibm-apiconnect"
  http_method => "post"
  codec => "json"
  content_type => "application/json"
  headers => ["x-nn-source-index", "<INDEX>", "x-nn-source-key", "<KEY>"]
  id => "noname-offload"
  ssl_certificate_authorities => "/etc/velox/external_certs/offload/cacert"
}
```

Combining offload configuration with filters

The following examples show how to use filters to control what data is offloaded.

Offload events from `catalog1` only:

```
   offload:
     enabled: true
     if  [catalog_name] == "catalog1" {
          http {
            url => "http://offloadhost.example.com/offload"
            http_method => "post"
            codec => "json"
            content_type => "application/json"
            id => "offload_http"
          }
        }
```

Offload to different locations based on catalog name:

```
offload:
  enabled: true
  output: |
    if  [catalog_name] == "catalog1" {
      http {
        url => "http://offloadhost.example.com /offload1"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload1_http"
      }
    }
    else if  [catalog_name] == "catalog2" {
      http {
        url => "http://offloadhost2.example.com /offload2"
        http_method => "post"
        codec => "json"
        content_type => "application/json"
        id => "offload2_http"
      }
    }
```

Use the catalog name as a variable in the offload location:

```
offload:
  enabled: true
  http {
    url => "http://offloadhost.example.com /%{[catalog_name]}"
    http_method => "post"
    codec => "json"
    content_type => "application/json"
    id => "offload_http"
  }
```

Do not offload events from `apic` and `sandbox` catalogs:

```
offload:
  enabled: true
  if  [catalog_name] !~ /sandbox|apic/ {
    http {
      url => "http://offloadhost.example.com /%{[catalog_name]}"
      http_method => "post"
      codec => "json"
      content_type => "application/json"
      id => "offload_http"
    }
  }
```

Using offloaded analytics data to train AI models

How to use API Connect analytics data to train AI models.

API Connect analytics holds full information on all of your API events. This information can prove invaluable in training AI models by using machine learning. Possible use cases include training data for automated tests or for security anomaly detection.

IBM Cloud Pak for Data (CP4D) includes the means to train AI models with data from various data sources. It is not possible for CP4D to pull data directly from API Connect's internal analytics database because the database is secured with no external access. However, you can offload your analytics data to an external database, and then configure CP4D to load data from that external database. You can use any external database that both API Connect analytics offload and CP4D support. The simplest option might be an external ELK stack (Elasticsearch Logstash Kibana). A list of supported data sources for CP4D can be found here: https://www.ibm.com/docs/en/cloud-paks/cp-data/latest?topic=data-supported-sources.

Documentation on how to configure Cloud Pak for Data to access Elasticsearch can be found here: https://www.ibm.com/docs/en/cloud-paks/cp-data/latest?topic=catalogs-elasticsearch-connection.

When your analytics data is offloaded to an external database, you can use it to train your AI model. Here is an example of how to create and train an AI model in CP4D using IBM Watson Machine Learning: https://www.ibm.com/docs/en/cloud-paks/cp-data/latest?topic=deploying-managing-ai-assets.

Troubleshooting offload

Checks to help determine the cause of offload failure.

- Check the ingestion pod logs for error messages that are related to offload:
  - Kubernetes, OpenShift, and Cloud Pak for Integration:

    ```
    kubectl -n <namespace> logs <analytics ingestion pod name>
    ```

  - VMware: Find the ingestion pod logs in Gathering logs for a VMware environment.

  A common problem is TLS handshaking errors, such as untrusted or expired certificates.
- Check logs on your third-party system.
- Verify that the specified URL or the host and port values for the target system are accurate.
- Verify that the server or cluster of servers in the target system are running and reachable. If applicable, also verify that firewall settings are not causing connection problems.
- If offloading data to an Elasticsearch or Kafka target, verify that your configuration settings are valid; for example, the user credentials for authenticating to Elasticsearch or the Kafka topic name.

Disable local storage

If you are offloading you API event data to a third-party system, you can disable local storage to reduce subsystem resource requirements.

The analytics subsystem provides a complete solution for routing, storing, and viewing the analytics data. However, you might not need the internal storage and viewing solution if you are using a third-party system for data offloading. In this scenario, you can greatly reduce your CPU and memory costs by disabling the internal storage and viewing components.

Important:
If you disable local storage, the analytics views in your API Connect UIs remain accessible, but are empty.

Configure local storage disablement before installation. It is not possible to disable or re-enable internal storage after installation.

You can enable offload of analytics data to third-party systems after installation.

Disabling local storage on Kubernetes, OpenShift and Cloud Pak for Integration

When you disable internal storage for analytics data, the following microservices are disabled:

- `osinit`
- `storage`

During the API Connect installation process, set `storage.enabled` to false in the CR:

```
storage:
  enabled: false
```

The relevant installation pages for each platform are:

- Kubernetes analytics install
- OpenShift analytics subsystem install
- OpenShift top-level CR install.
- Cloud Pak for Integration: Use the YAML tab to edit the CR before installation, see *API management deployment* in https://www.ibm.com/docs/en/cloud-paks/cp-integration.

Disabling local storage on VMware

During installation of the analytics subsystem, create the analytics extra-values file, and insert:

```
storage:
  enabled: false
```

Do not create your analytics subsystem ISO files before the `extra-values-file` property is set:

```
apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
```

Shutting down the analytics subsystem

Disable the analytics service and shutdown the analytics subsystem.

About this task

This operation requires the use of the analytics CLI.

Disabling the analytics subsystem stops the collection and storage of data and makes the subsystem inaccessible.

Procedure

1. Backup your analytics data as explained in Backup, restore, and disaster recovery.
2. Setup the toolkit CLI so you can run analytics CLI operations: Analytics CLI.
3. Optional: Disable shard allocation.
   Disable shard allocation to prevent new shards from being created for replication when a node is unavailable. For more information about shards, see OpenSearch nodes, indices, shards, and replicas.

   a. Create a JSON file called `analytics_persistent_settings.json` and insert the contents:

   ```
   {"persistent":{"cluster.routing.allocation.enable":"none"}}
   ```

   b. Run the following analytics CLI command:

   ```
   apic --mode analytics clustermgmt:putSettings --server <platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
   ```

   where:
   - *<platform api endpoint>* is the platform API endpoint.
   - *<analytics service name>* is the name of analytics service as configured in the Cloud Manager UI.

   Expected output:

   ```
   {"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":{"enable":"none"}}}},"transient":{}}
   ```

4. Optional: Run a synced flush of storage.
   The synced flush operation flushes all current index operations synchronously and writes all in-flight transactions to disk.

   ```
   apic --mode analytics clustermgmt:postFlush --server <platform api endpoint> --analytics-service <analytics service name> --format json
   ```

   where:
   - *<platform api endpoint>* is the platform API endpoint.
   - *<analytics service name>* is the name of analytics service as configured in the Cloud Manager UI.

   When you flush storage, the response looks like the following example, with the *failed* count set to zero:

   ```
   {
       "_shards": {
           "total": 16,
           "successful": 6,
           "failed": 0
   ```

```
    }
}
```

This operation occasionally fails, but it is safe to rerun it multiple times until it passes and the "failed" count is zero.

5. Unassociate the analytics service from all gateway services.
   a. In Cloud Manager, click Topology.
   b. In the section for the Availability Zone that contains the analytics service, locate the Gateway service that the analytics service is associated with.
   c. Click ⋮ and select Unassociate.

   For more information, see Associating an analytics service with a gateway service.

6. Unregister the analytics service from Cloud Manager:
   a. In Cloud Manager, click Topology.
   b. In the section for the Availability Zone that contains the analytics service, locate the analytics service and click Delete.

   For more information, see Registering an analytics service.

7. **Kubernetes, OpenShift, and Cloud Pak for Integration only:** Set replicas to zero for the analytics pods by editing the Deployments and StatefulSets that manage them: `director`, `mtls-gw`, `ingestion`, and `storage`:

   ```
   kubectl -n <namespace> edit deploy analytics-director
   kubectl -n <namespace> edit deploy analytics-mtls-gw
   kubectl -n <namespace> edit sts analytics-ingestion
   kubectl -n <namespace> edit sts analytics-storage
   ```

   Note: The Deployments and StatefulSets on Cloud Pak for Integration and OpenShift with the top-level CR have the name format *<instance name>-a7s-...*
   Set *replicas* to zero:

   ```
   spec:
     ...
     replicas: 0
     ...
   ```

   The analytics `director`, `mtls-gw`, `ingestion`, and `storage` pods all terminate.
8. **VMware only:** Shutdown your analytics VMs from your VMware console.

Reenabling the analytics subsystem

9. **Kubernetes, OpenShift, and Cloud Pak for Integration only:** Repeat step 7 but in reverse order and setting the *replicas* value back to what it was before.
   The pods all start:

   ```
   kubectl get deploy -n <namespace>
   analytics-director                          1/1     1            1           7h10m
   analytics-mtls-gw                           1/1     1            1           7h6m

   kubectl get sts -n <namespace>
   analytics-ingestion   1/1     7h10m
   analytics-storage     1/1     7h12m
   ```

10. **VMware only:** Start your analytics VMs from your VMware console.
11. Reregister the analytics service: Registering an analytics service.
12. Reassociate your analytics service with your gateways Associate new analytics service(s) with gateway(s).
13. Enable shard allocation by following these steps:
    Note: If you did not disable shard allocation in step 3, skip this step.
    a. Create a JSON file called `analytics_persistent_settings.json` and insert the contents:

    ```
    {"persistent":{"cluster.routing.allocation.enable":"all"}}
    ```

    b. Run the following analytics CLI command:

    ```
    apic --mode analytics clustermgmt:putSettings --server <platform api endpoint> --analytics-service <analytics service name> --format json analytics_persistent_settings.json
    ```

    where:
    - *<platform api endpoint>* is the platform API endpoint.
    - *<analytics service name>* is the name of analytics service as configured in the Cloud Manager UI.

    When you enable shard allocation, the response looks like the following example:

    ```
    {"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":{"enable":"all"}}}},"transient":{}}
    ```

Restarting analytics pods

How to restart the analytics pods in your API Connect deployment.

About this task

Certain scenarios might require you to restart your analytics subsystem pods. To restart your analytics pods, follow the steps in this topic.

Procedure

1. **VMware users only:** Log in to your analytics VM and switch to the root user:

   ```
   ssh apicadm@<vm hostname>

   sudo -i
   ```

2. Restart the analytics pods.
   - **Kubernetes and OpenShift:** Run the following command:

     ```
     kubectl -n <namespace> delete pod -l app.kubernetes.io/instance=<analytics CR>
     ```

   - **VMware:** Run the following command:

     ```
     kubectl delete pods --all
     ```

3. Verify that all analytics pods restart successfully:

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

   where *<namespace>* is the namespace where the analytics subsystem is installed. VMware users can omit `-n <namespace>`.

Change deployment profile

Change your deployment profile to switch between one replica and three replica deployments, or to change the computing resources available for your subsystem.

- **Change deployment profile on Kubernetes and OpenShift**
  Update your analytics CR to change your deployment profile.
- **Change deployment profile on VMware**
  Use the **apicup** command and deploy new VMs to change your deployment profile.

Change deployment profile on Kubernetes and OpenShift

Update your analytics CR to change your deployment profile.

Note: This topic covers Kubernetes and OpenShift individual subsystem installations. For other platforms see:

- Changing deployment profile on Cloud Pak for Integration
- Changing deployment profile on OpenShift, top-level CR

Note: If you have a three replica profile, and want to change to one replica, you must first change your analytics configuration to shared storage. Dedicated storage is not supported on the one replica profile.

1. Verify that subsystem is in `Running` state.

   ```
   kubectl -n <namespace> get a7s
   ```

2. Edit the analytics CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

   Change `spec.profile` value to the new profile that you want to use: Analytics deployment profiles.

3. Monitor you analytics pods to confirm the expected number of replicas are running:

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

4. Verify that subsystem is in `Running` state.

   ```
   kubectl -n <namespace> get a7s
   ```

Change deployment profile on VMware

Use the **apicup** command and deploy new VMs to change your deployment profile.

For information about the available deployment profiles, see Planning your deployment topology and profiles.
Note:
The procedures to change deployment profile do not permit the endpoints to be changed.

If you want to change a one replica deployment to a three replica deployment, and you used the hostname of your original VM in your endpoints, you cannot change these endpoints. In this scenario, you must configure load-balancers and logical DNS records so that your original VM hostname routes to the new VMs of your three replica deployment.

If you want to change endpoints, you can use the form-factor migration process.

Moving to a profile that has the same number of VMs

Complete the following steps to change your deployment profile:

1. Take backups of your analytics configuration and database: VMware analytics backup.
2. Shutdown your analytics VMs.
3. Update the CPU and memory resources for your analytics VMs in your VMware console, as required for your new profile.
4. Restart your VMs.
5. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem name> deployment-profile=<new profile>
```

6. Apply the new profile:

```
apicup subsys install <subsystem name>
```

If for any reason you are not able to change the CPU or memory on your VMs, redeploy them following the disaster recovery process: Recovering from a disaster.

Moving from a one replica to a three replica deployment profile

1. Take backups of your analytics configuration and database: VMware analytics backup.
   Note: You do not need to take VMware snapshots, since your existing VM is deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

```
apicup subsys set <analytics subsystem name> deployment-profile=<three replica profile>
```

3. Define the hostname and network interfaces for the two new analytics VMs. Use the `apicup hosts create` and `apicup iface create` commands. For more information, see Configuring the Analytics subsystem.
4. Shutdown your existing analytics VM.
5. Follow the disaster recovery steps: Recovering from a disaster, but note that you must create a new ISO file for your original single analytics VM. Do not attempt to reuse the previous ISO file in your new three replica deployment.
6. When disaster recovery steps are complete and your new deployment is verified, delete your original VM (which you shutdown in step 4).

Moving from a three replica to a one replica deployment profile

1. Take backups of your analytics configuration and database: VMware analytics backup.
   Note: You do not need to take VMware snapshots, since your existing VMs are deleted and replaced when you change deployment profile.
2. In your project directory, update your deployment profile:

```
apicup subsys set <subsystem name> deployment-profile=<one replica profile>
```

3. Delete two of your three VMs from your configuration with the `apicup hosts delete` command.
4. Shutdown your existing API Connect VMs.
5. Follow the disaster recovery steps: Recovering from a disaster, but note that you must create a new ISO file for your analytics VM. Do not attempt to reuse a previous ISO file in your new one replica deployment.
6. When disaster recovery steps are complete and your new deployment is verified, delete your original VMs (which you shutdown in step 4).

Performance tuning with ingestion pipeline properties

Fine tune your ingestion pipeline settings for optimal performance.

Table 1.

Field	Type	Default value	Description	Comments
`ingestion: pipelineWorkers`	4 <= integer <= 128	24	Number of pipeline workers.	More workers mean more CPU and memory requirements, but gives you the benefit of extra threads to pull data off the ingestion HTTP interface and process it for storage. Increasing the number of pipeline workers excessively results in diminishing returns. `pipelineWorkers` should usually be a multiple of the number of cores of your hardware. Adjust with caution.
`ingestion: pipelineBatchSize`	10 <= integer <= 10000	250	Batch size of a pipeline worker.	Pulling more or less data off the HTTP interface per worker can have benefits. Recommended values are `125`, `250`, and `500`. Increasing the value requires more memory.
`ingestion: pipelineBatchDelay`	10 <= integer <= 5000	50	Batch delay of a pipeline worker.	How long to wait for the batch size to fill before leaving the HTTP interface to process for storage. Having threads wait too long causes idle threads in your system; however, setting this value too low decreases performance. Adjust with caution.
`ingestion: configReloadAutomatic`	Boolean	true	Whether to automatically reload the pipeline on configuration changes.	Typically, when the underlying files for the pipeline are modified, the pipeline process automatically reloads and picks up the changes. However, in Kubernetes environments where the pipeline is mapped to ConfigMaps, this is unreliable. Ingestion pod restarts are usually required.
`ingestion: pipelineUnsafeShutdown`	Boolean	false	Whether to allow unsafe shutdown of the pipeline.	Typically, the pipeline attempts to empty all workers' batches before shutdown to prevent data loss. You might need to enable this setting if the pipeline is badly configured and refuses to respond to shut down requests (uncommon). Adjust with caution.

Customize your analytics data

How to customize the data stored in your API event records.

- **Activity log policy**
  Configure the activity log policy to manage the level of detail of the data stored in API events.
- **Customizing data with a gateway script or set-variable operation**
  Customize your API event records on the gateway before they are sent to your analytics subsystem.

- **Customizing data with ingestion filters**
  Customize you API event records as they are received from the gateway.

Activity log policy

Configure the activity log policy to manage the level of detail of the data stored in API events.

When you configure an API in the API Manager UI, you choose what data is stored in the API event records for calls to that API with the **activity log** property. The **activity log** property has separate settings for successful API calls, and for failed API calls. The activity log options are:

- none: API events are not logged.
- activity: Logs the API resource URI. **activity** is the default log setting for successful API calls.
- header: Logs the API resource URI and HTTP headers.
- payload: Logs API resource URI, HTTP headers, and request and response payloads. **payload** is the default log setting for failed API calls, since the response payload might have useful information on the cause of the failure.

Note:
If your gateway is configured to authenticate all API calls with an OAuth provider, the data that is related to the OAuth call is only included in the API event data when the OAuth call fails. Successful OAuth authentication is not recorded in the API event data.

Defining log setting in the API Manager UI

1. In the API Manager UI Develop view, select the API that you want to work with. If you are not familiar with the API Manager UI Develop view, see Working with API definitions.
2. Select the Gateway tab, expand Gateway and portal settings, then select Activity Log.
3. Select Enabled, then select the preferred log setting for Success Content and Error Content:
   - none: API events are not logged.
   - activity: Logs the API resource URI. **activity** is the default log setting for successful API calls.
   - header: Logs the API resource URI and HTTP headers.
   - payload: Logs API resource URI, HTTP headers, and request and response payloads. **payload** is the default log setting for failed API calls, since the response payload might have useful information on the cause of the failure.
4. Click Save to save your changes.

Defining log setting in source for DataPower® API Gateway

Use the **activity-log** extension to configure your logging preferences.

```
activity-log:
  success-content: <success log setting: none, activity, header, payload>
  error-content: <failure log setting: none, activity, header, payload>
  enabled: <if activity logging is enabled: true or false>
```

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
  success-content: <success log setting: none, activity, header, payload>
  error-content: <failure log setting: none, activity, header, payload>
  enabled: <if activity logging is enabled: true or false>
buffering: true
```

Examples:

No logging of successful calls, header logging for failed calls

```
activity-log:
    success-content: none
    error-content: header
    enabled: true
```

Disable activity logging

```
activity-log:
    enabled: false
```

Defining log setting in source for DataPower Gateway (v5 compatible)

The activity-log policy has the following format:

```
- activity-log:
  version: <version>
  title: <title>
  description: <description>
  content: <success log setting: none, activity, header, payload>
  error-content: <failure log setting: none, activity, header, payload>
```

where:

- *<version>* is the policy version number. Set version to 1.0.0.
- *<title>* is a title for the policy.
- *<description>* is a description for the policy.

Apply this policy by adding an assembly extension with an execute field to your OpenAPI definition file.

Note: If payload logging is enabled, for the gateway to capture payloads buffering must also be enabled.

```
activity-log:
  success-content: activity_to_log_if_call_successful
  error-content: activity_to_log_if_call_unsuccessful
  enabled: is_activity_logging_enabled
buffering: true
```

Examples:

Maintain default log behavior

Do not specify **content** or **error-content**:

```
- activity-log:
  version: 1.0.0
  title: default activity logging
```

No logging of successful calls, activity logging for failed calls

```
- activity-log:
  version: 1.0.0
  title: no logging for successful calls
  content: none
  error-content: activity
```

Customizing data with a gateway script or set-variable operation

Customize your API event records on the gateway before they are sent to your analytics subsystem.

Before you begin

Ensure that you are familiar with the API Manager UI, and working with API definitions.
Note: This feature is not available with DataPower® Gateway (v5 compatible).

About this task

During API execution, the activity data is stored in the **log** context variable, which populates the API event record on completion of the API execution. For more information about **log** context variables, see API activity logging context variables.

The analytics subsystem stores analytics data as individual API event records. The API event record fields are documented here: API event record field reference. It is possible to customize the API event records for specific APIs. You can modify the contents of existing API event fields, and add new custom fields by using a gateway script or the **set-variable** operation.
Note: If you want to make updates to all incoming API event data, rather than specific APIs, you can use ingestion filters instead.

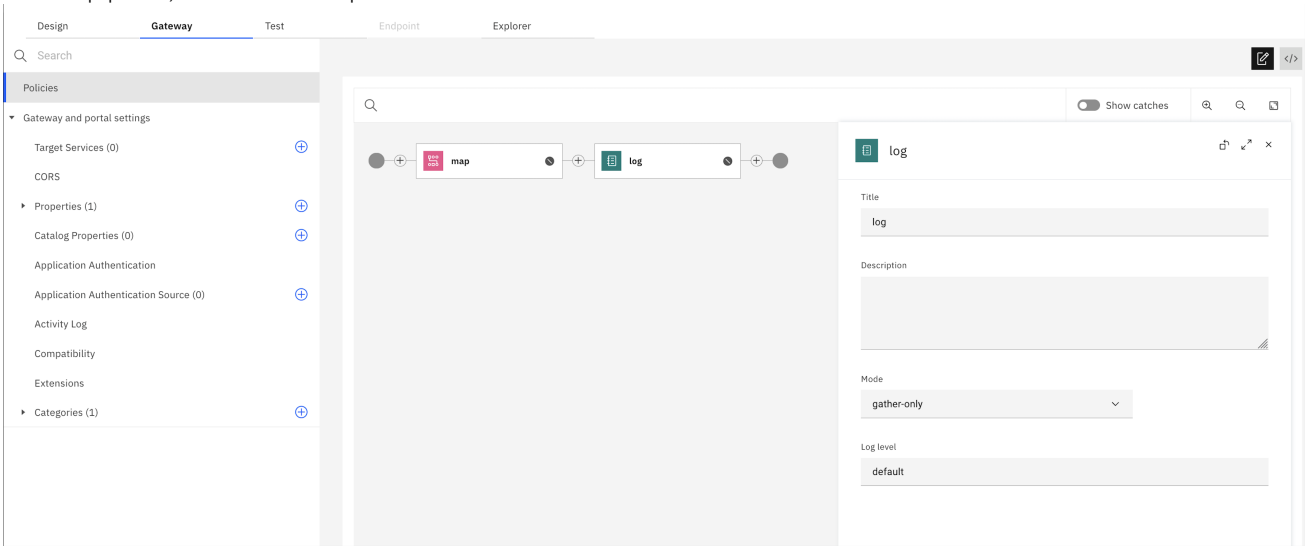
Procedure

1. In the API Manager UI, click the Develop APIs and products tile, select the APIs tab, click Add, select API.
2. Select New OpenAPI, give the API the title of "Hello World", and click Next. Click Next again to accept the security defaults, and then click Edit API.
3. Select the YAML source view </>. Replace the generated YAML with the following YAML and click Save:

```
swagger: '2.0'
info:
  title: Hello World
  x-ibm-name: hello-world-rest
  version: 1.0.0
schemes:
  - https
basePath: /hello-world
security:
  - {}
securityDefinitions:
  clientID:
    type: apiKey
    in: header
    name: X-IBM-Client-Id
x-ibm-configuration:
  phase: realized
  testable: true
  enforced: true
  properties:
    target-url:
      value: http://example.com/operation-name
      description: The URL of the target service
      encoded: false
  cors:
    enabled: true
  activity-log:
    enabled: true
    error-content: header
    success-content: activity
  application-authentication:
    certificate: false
  assembly:
    execute:
      - map:
```

```
        version: 2.0.0
        title: map
        inputs:
          name:
            schema:
              type: string
            variable: request.parameters.name
        outputs:
          response:
            schema:
              $ref: '#/definitions/Result'
            variable: message.body
        actions:
          - set: response.hello
            from: name
            value: '''Hello '' +$(name)'
    catch: []
    finally: []
  gateway: datapower-api-gateway
  categories:
    - /tests
  type: rest
definitions:
  Result:
    type: object
    additionalProperties: false
    properties:
      hello:
        type: string
paths:
  /hello:
    get:
      parameters:
        - name: name
          in: query
          required: false
          type: string
      responses:
        '200':
          description: OK Response
          schema:
            $ref: '#/definitions/Result'
    parameters: []
```

4. Click the Form icon to switch to the Form view and select the Gateway tab.
5. Ensure that Policies is selected.
6. On the canvas, click the plus icon at the last position in the flow. The palette listing the available elements to build the assembly flow appears.
7. From the palette, add the Log policy element onto the canvas by clicking it. Set the log policy Mode drop-down to Gather-only. This step is so that the "log" context variable is populated, for use in the next step.

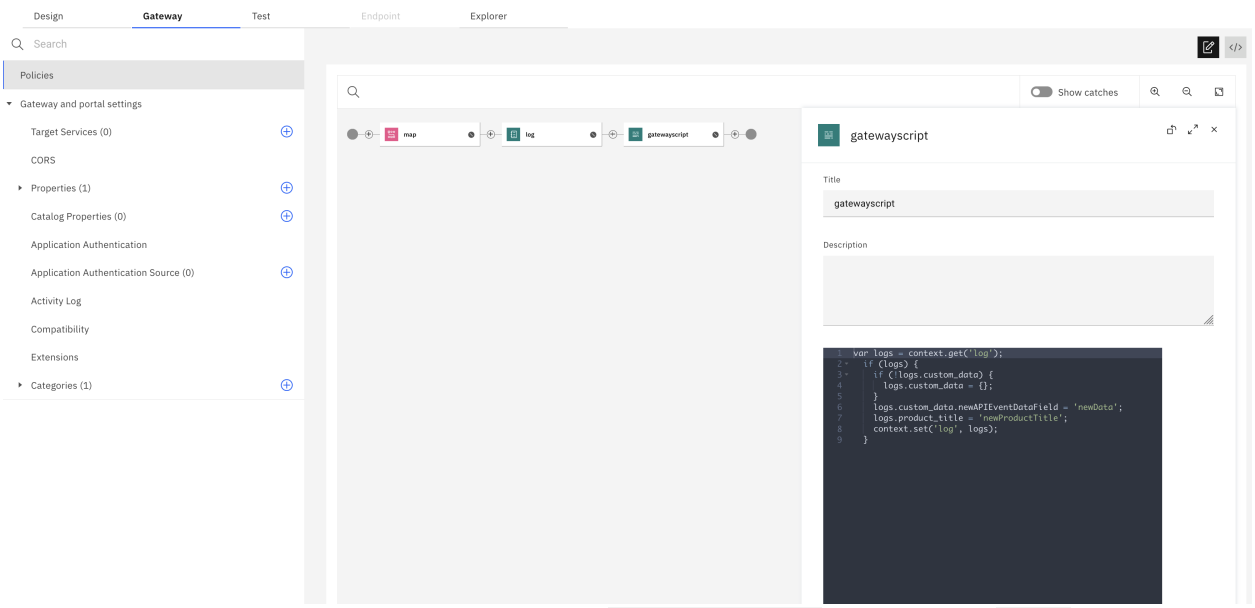


8. Modify API event data or add custom data either with a gateway script or by using the set-variable operation.

   Gateway script method

   Add a gateway script policy to the assembly flow and paste in the following script code:

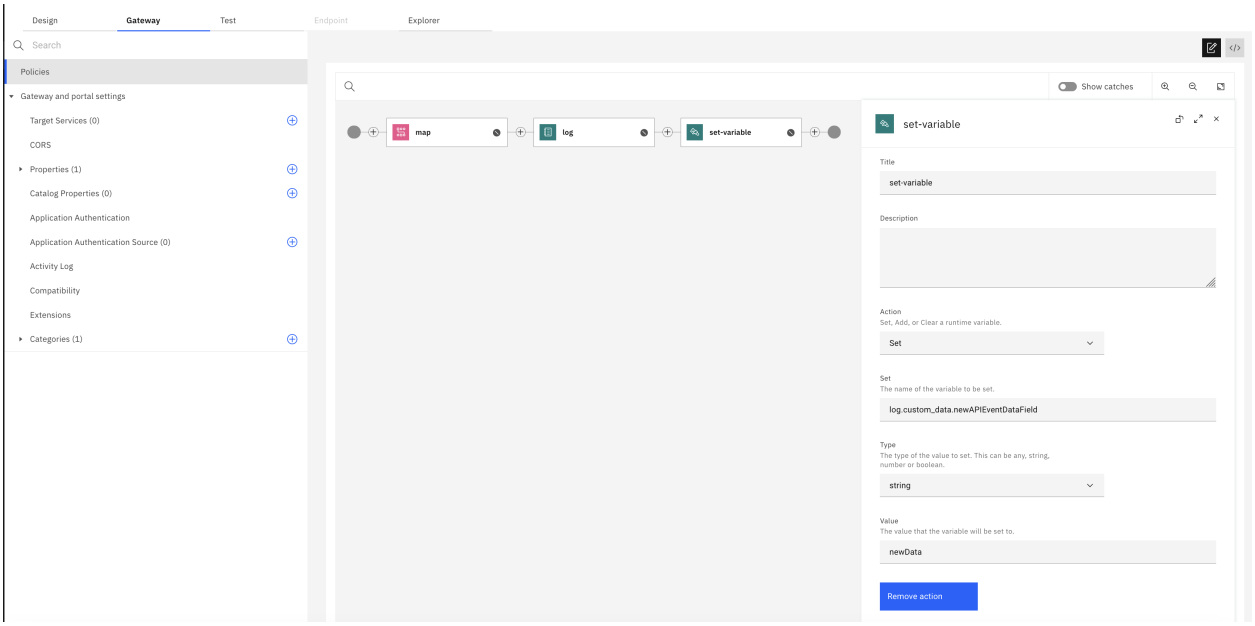
```
var logs = context.get('log');
  if (logs) {
    if (!logs.custom_data) {
      logs.custom_data = {};
    }
    logs.custom_data.newAPIEventDataField = 'newData';
    logs.product_title = 'newProductTitle';
    context.set('log', logs);
  }
```



In this example, the new field that is added to the API event record is called **`newAPIEventDataField`** and its value is set to **`newData`**. The value of the existing field **`product_title`** is changed to **`newProductTitle`**.

Set-variable method

If you cannot or do not want to use a gateway script, then you can use a set-variable policy instead. Add a set-variable policy to the assembly flow, and set it as follows:



In this example, the new field that is added to the API event record is called **`newAPIEventDataField`** and its value is set to **`newData`**

The YAML code for set-variable is:

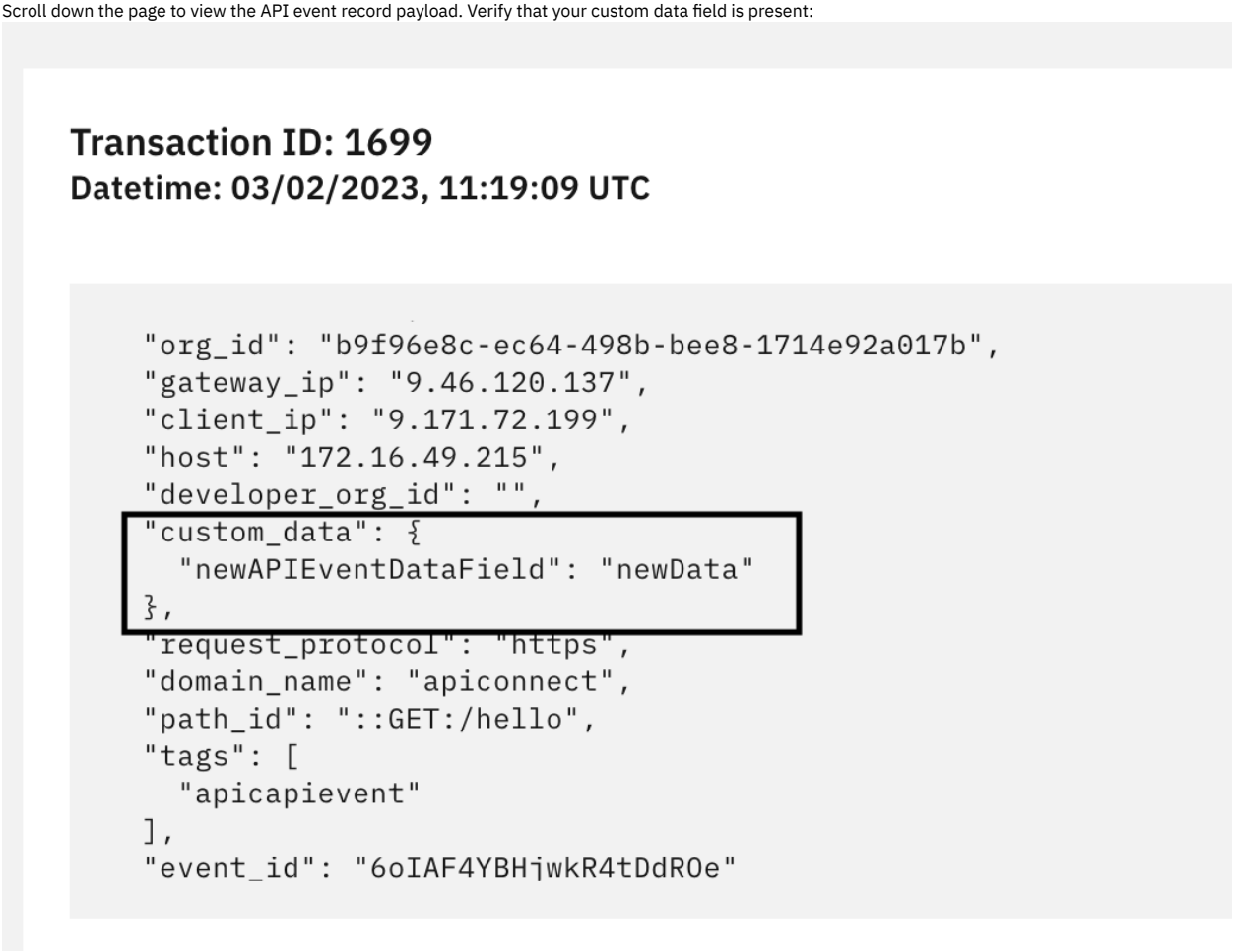
```
    - set-variable:
        version: 2.0.0
        title: set-variable
        actions:
          - set: log.custom_data.newAPIEventDataField
            value: newData
            type: string
          - set: log.product_title
            value: newProductTitle
            type: string
```

Note: **`context.log.custom_data`** supports the addition of primitive typed values such as strings and integers, no limit is enforced on how many you can add. Add data only to the **`context.log.custom_data`** object. Do not add data to **`context.log`**.

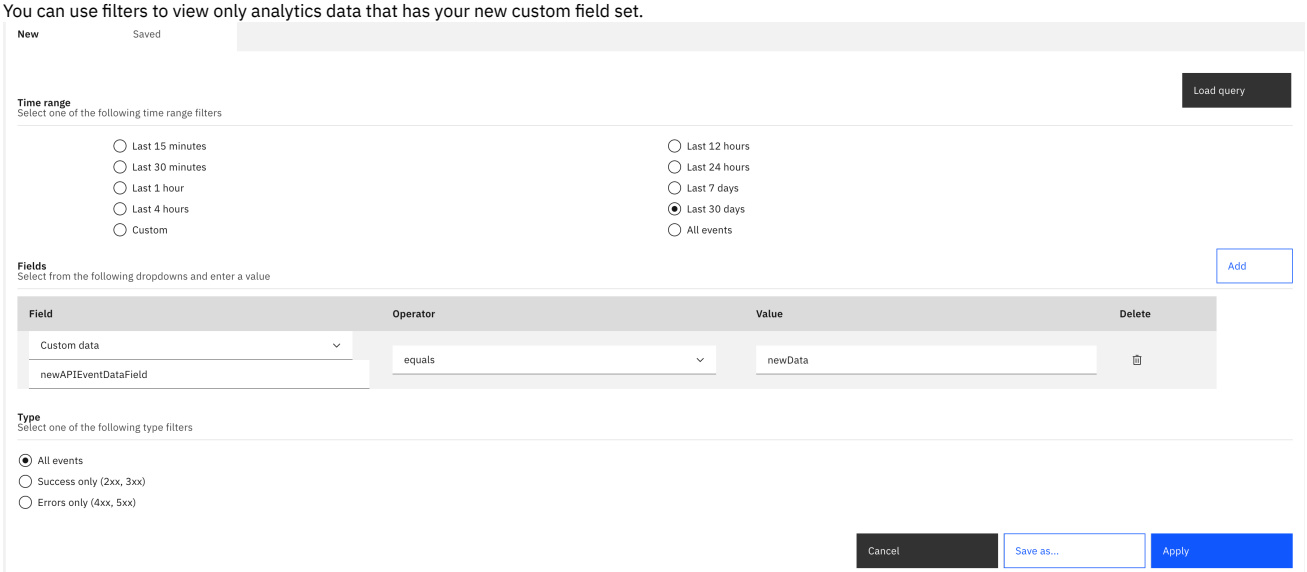
The type of existing fields cannot be changed. For example, **`product_title`** is a string and cannot be changed to an integer. Do not attempt to assign a different type to an API event field.

9. Publish your "Hello World" API to a catalog.
10. Make a test call to the API and verify that the new custom data is set.
    a. From the Analytics view, select Discover and click the API event record corresponding to your test call:

b. Scroll down the page to view the API event record payload. Verify that your custom data field is present:



11. You can use filters to view only analytics data that has your new custom field set.



12. To use new data with REST API calls, use the custom_data query parameter:

```
/cloud/events?custom_data[newAPIEventDataField]=newData
```

Customizing data with ingestion filters

Customize you API event records as they are received from the gateway.

Create ingestion filters to customize the analytics API event records as they are received from the gateway and before they are sent on to internal storage or offloaded to a third-party system. Filters enable you to add new fields, modify existing fields, or remove fields from the API event record. Filters are applied to all incoming API events, but you can include a conditional check in your filter syntax if you want a filter to apply to specific APIs only.

Ingestion filters use Logstash plug-ins. For more information about Logstash plug-ins, see the Logstash documentation.

All fields in the API event record are modifiable with ingestion filters, but to avoid problems you must adhere to the following restrictions:

- Do not modify the following fields:
  - org_id
  - catalog_id
  - space_id
  - developer_org_id
  - datetime
  - @timestamp
- Do not change the data type of existing fields. Create a new field that has the type you want and copy the contents to this field.
- If you want to sanitize data, it is more efficient to delete fields entirely than delete their contents.

To review all the fields in the API event record, see API event record field reference.

For examples of ingestion filters, see Example ingestion filters.

The steps for configuring ingestion filters depend on your platform:

- **Kubernetes and OpenShift individual subsystem installations**
  Define ingestion filters to customize API event data before it is stored or offloaded.
- **Cloud Pak for Integration and OpenShift top-level CR**
  Define ingestion filters to customize API event data before it is stored or offloaded.
- **VMware**
  Define ingestion filters to customize API event data before it is stored or offloaded.

Kubernetes and OpenShift individual subsystem installations

Define ingestion filters to customize API event data before it is stored or offloaded.

Procedure

1. Edit your `AnalyticsCluster` CR:

   ```
   kubectl -n <namespace> edit a7s
   ```

2. Filters for API events that are offloaded to a third-party system are defined in the `external.offload` section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the `ingestion` section.

   Ingestion filters - for API event data stored by API Connect
   Add the filter definition to the `ingestion` section:

   ```
   ingestion:
     ...
     filter: |
       if [api_name] == "test-api" {
         mutate {
           remove_field => ["query_string"]
         }
       }
   ```

   In this example, a conditional filter is used to apply the change to API event records where the API name is `test-api`.

   Offload filters - for API event data that is sent to a third-party system.
   Add the filter definition to the `external.offload` section:

   ```
   external:
     offload:
       filter: |
         mutate {
           remove_field => ["query_string"]
         }
   ```

   In this example, no conditional filter is used; the change is applied to all APIs.

   For more examples, see Example ingestion filters.

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting `geoIPEnabled: true` in the ingestion section:

   ```
   ingestion:
     ...
     geoIPEnabled: true
   ```

   By default the `client_geoip` and `gateway_geoip` fields are not included in the analytics data. You do not need to add `geoIPEnabled` to the `external.offload` section. Enabling geoIP in the `ingestion` section enables it for both data routes.

   Note:
   To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: NGINX Configuration "use-forwarded-headers".

   The geoIP feature is useful with internet routable IP addresses only. For instance, 192.168.x.x and 10.x.x.x IP addresses, being internal private IP addresses, cannot be resolved to a geographical location.

4. Save the changes. Updates to filters take effect after approximately 30 seconds.

Cloud Pak for Integration and OpenShift top-level CR

Define ingestion filters to customize API event data before it is stored or offloaded.

About this task

Procedure

1. Edit your **APIConnectCluster** CR:

   ```
   oc -n <namespace> edit APIConnectCluster
   ```

2. In the **analytics.spec** section, add your ingestion filter.
   Filters for API events that are offloaded to a third-party system are defined in the **external.offload** section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the **ingestion** section.

   Ingestion filters - for API event data stored by API Connect
   Add the filter definition to the **ingestion** section:

   ```
   ingestion:
     ...
     filter: |
       if [api_name] == "test-api" {
         mutate {
           remove_field => ["query_string"]
         }
       }
   ```

   In this example, a conditional filter is used to apply the change to API event records where the API name is **test-api**.
   Offload filters - for API event data that is sent to a third-party system.
   Add the filter definition to the **external.offload** section:

   ```
   external:
     offload:
       filter: |
         mutate {
           remove_field => ["query_string"]
         }
   ```

   In this example, no conditional filter is used; the change is applied to all APIs.

   For more examples, see Example ingestion filters.

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting **geoIPEnabled: true** in the ingestion section:

   ```
   ingestion:
     ...
     geoIPEnabled: true
   ```

   By default the **client_geoip** and **gateway_geoip** fields are not included in the analytics data. You do not need to add **geoIPEnabled** to the **external.offload** section. Enabling geoIP in the **ingestion** section enables it for both data routes.

   Note:
   To ensure the value in the **client_geoip** field is accurate, the gateway must receive the **X-Forwarded-For** header in all API calls. Check with the administrator of your deployment environment to ensure that the **X-Forwarded-For** header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the **X-Forwarded-For** header: NGINX Configuration "use-forwarded-headers".

   The geoIP feature is useful with internet routable IP addresses only. For instance, 192.168.x.x and 10.x.x.x IP addresses, being internal private IP addresses, cannot be resolved to a geographical location.

4. Save the changes. Updates to filters take effect after approximately 30 seconds.

VMware

Define ingestion filters to customize API event data before it is stored or offloaded.

About this task

Procedure

1. From your project directory, open the analytics-extra-values.yaml file for editing. If this file does not exist, then create it: Analytics extra-values file.
2. Inside the **spec** object of your analytics-extra-values.yaml file, define your ingestion filters.
   Filters for API events that are offloaded to a third-party system are defined in the **external.offload** section. Filters for API events that are stored in the API Connect OpenSearch database are defined in the **ingestion** section.

   Ingestion filters - for API event data stored by API Connect
   Add the filter definition to the **ingestion** section:

```
ingestion:
  ...
  filter: |
    if [api_name] == "test-api" {
      mutate {
        remove_field => ["query_string"]
      }
    }
```

In this example, a conditional filter is used to apply the change to API event records where the API name is `test-api`.

Offload filters - for API event data that is sent to a third-party system.

Add the filter definition to the `external.offload` section:

```
external:
  offload:
    filter: |
      mutate {
        remove_field => ["query_string"]
      }
```

In this example, no conditional filter is used; the change is applied to all APIs.

For more examples, see Example ingestion filters.

3. Optional: Include geoIP fields in your analytics data, for both stored and offloaded data, by inserting `geoIPEnabled: true` in the ingestion section:

```
ingestion:
    ...
    geoIPEnabled: true
```

By default the `client_geoip` and `gateway_geoip` fields are not included in the analytics data. You do not need to add `geoIPEnabled` to the `external.offload` section. Enabling geoIP in the `ingestion` section enables it for both data routes.

Note:
To ensure the value in the `client_geoip` field is accurate, the gateway must receive the `X-Forwarded-For` header in all API calls. Check with the administrator of your deployment environment to ensure that the `X-Forwarded-For` header is passed to your gateways. For example, in a Kubernetes environment where NINGX ingresses are used, configure the NGINX ingress that is used by your gateway to use the `X-Forwarded-For` header: NGINX Configuration "use-forwarded-headers".

The geoIP feature is useful with internet routable IP addresses only. For instance, 192.168.x.x and 10.x.x.x IP addresses, being internal private IP addresses, cannot be resolved to a geographical location.

4. Ensure that your `extra-values-file` property is set to point to your analytics-extra-values.yaml file:

```
apicup subsys set <analytics subsystem> extra-values-file analytics-extra-values.yaml
```

5. Apply the analytics-extra-values.yaml to your analytics subsystem:

```
apicup subsys install <analytics subsystem>
```

Example ingestion filters

Review example ingestion filters to see how to add, modify, and delete fields from API event records.

You can use these example filters in the `ingestion.filter` and the `external.offload.filter` configurations. For more information about the filter syntax, see the Logstash documentation: https://www.elastic.co/guide/en/logstash/current/filter-plugins.html.

Add a field

You can add custom data to API event data by adding a field.

To avoid naming conflicts with future or current analytics fields, include a prefix that ensures your new field name is unique. For example, instead of naming the field `employee_num` you might name it `x_mycompany_employee_num`.

The following example copies the contents of the `X-Employee-Num` field from the request header and adds it to the `x_mycompany_employee_num` field.

```
if [request_http_headers] {
  ruby {
    code => "event.get('[request_http_headers]').collect {|i| event.set('[x_mycompany_employee_Num]', i['X-Employee-Num']) if
i.has_key?('X-Employee-Num')}"
  }
}
```

Modifying an existing field

Sometimes you don't want to remove a field entirely from your data, but you want to redact sensitive information such as IDs. You can modify the contents of a field and replace information with symbols or a message.

Remember: The following fields must not be modified if the data is being written to internal analytics storage: `org_id`, `catalog_id`, `space_id`, `developer_org_id`, `datetime`, and `@timestamp`.
The following example replaces sensitive information in the `X-Employee-Name` and `X-Employee-ID` request headers with the string: `*******sanitized*******`.

```
if [request_http_headers] {
  ruby {
    code => "headers=['X-Employee-Name','X-Employee-ID']; newHeaders = event.get('[request_http_headers]').collect {|i|
```

```
headers.each {|header| i[header] = '********sanitized********' if i.has_key?(header)}; i}; event.set('[request_http_headers]',
newHeaders)"
  }
}
```

Removing an existing field

Use the mutate **remove_field** operation to delete a field. To remove multiple fields, delimit the field names with commas.

The following example removes multiple fields (**request_http_headers**, **response_http_headers**, **request_body**, **response_body**, and **query_string**) from the API event record:

```
mutate {
  remove_field => ["request_http_headers", "response_http_headers", "request_body", "response_body", "query_string"]
}
```

Drop API event records

If you want certain API events to not be stored at all, use the **drop** syntax.

Do not store events from the **apic** and **sandbox** catalogs:

```
if  [catalog_name] =~ /sandbox|apic/ {
  drop { }
}
```

Access your analytics data

How to view, analyze, and export your analytics data.

- **API Connect UIs**
  View, query, and export your analytics data from the API Connect UIs.
- **Counting total API calls for your analytics service**
  Count the API calls for the analytics subsystem and see the total categorized by response status.
- **Export your analytics data**
  Export your analytics data with either the toolkit CLI or the REST API.
- **Allowing API Connect cloud admin users to view provider organization analytics data**
  The owner of a provider organization can enable or disable the sharing of analytics data with API Connect cloud administrators.
- **Analytics CLI**
  How to access your analytics data with the toolkit CLI.
- **Analytics REST API**
  How to access your analytics data with the REST API.

API Connect UIs

View, query, and export your analytics data from the API Connect UIs.

- **Accessing analytics in the Cloud Manager UI**
  How to view, analyze, and export your analytics data from the Cloud Manager UI.
- **Accessing analytics in the API Manager UI**
  How to view, analyze, and export your analytics data from the API Manager UI.
- **Accessing analytics in the Developer Portal**
  How to view your analytics data from the Developer Portal.

Accessing analytics in the Cloud Manager UI

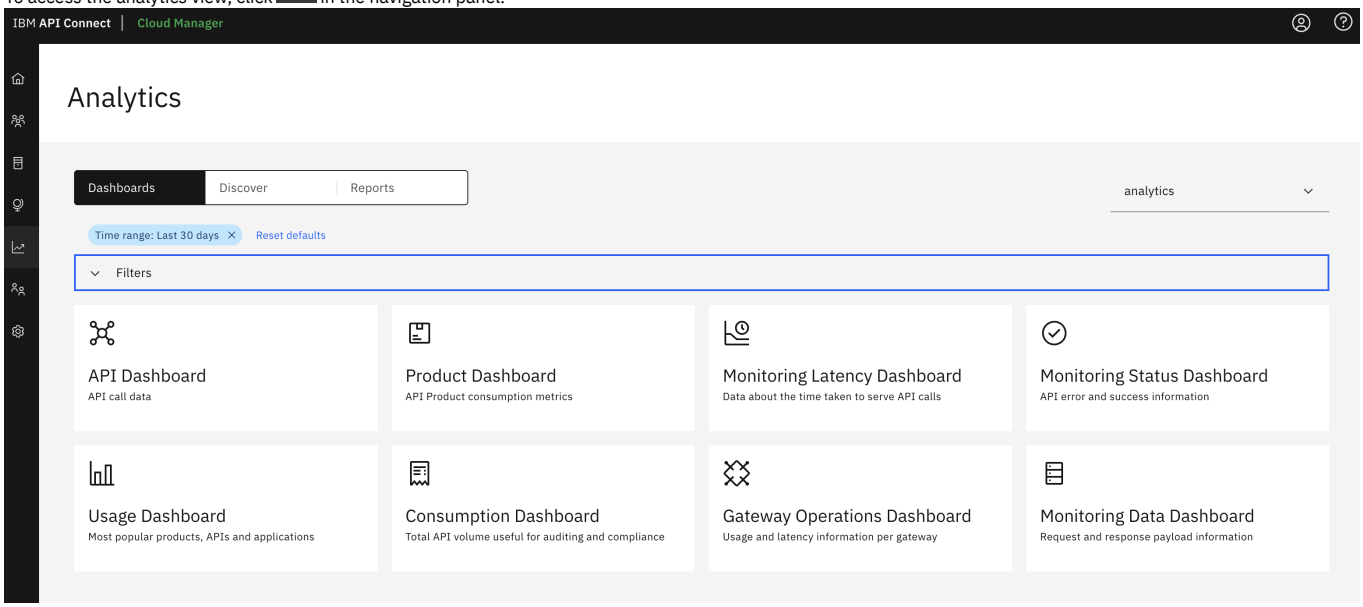
How to view, analyze, and export your analytics data from the Cloud Manager UI.

The Cloud Manager UI analytics view shows total API call volume data, and total call volumes per gateway.

The Cloud Manager UI analytics view also shows analytics data for provider organizations where the organization owner allows sharing of this data. For more information about analytics data sharing, see Allowing API Connect cloud admin users to view provider organization analytics data.
Note: If no provider organization owners enable data sharing with the cloud admin user, then the only analytics data you can see are on the following dashboards:

- Consumption Dashboard - contains charts that show total API consumption in your cloud.
- Gateway Operations Dashboard - contains charts that show API usage and latency by gateway.

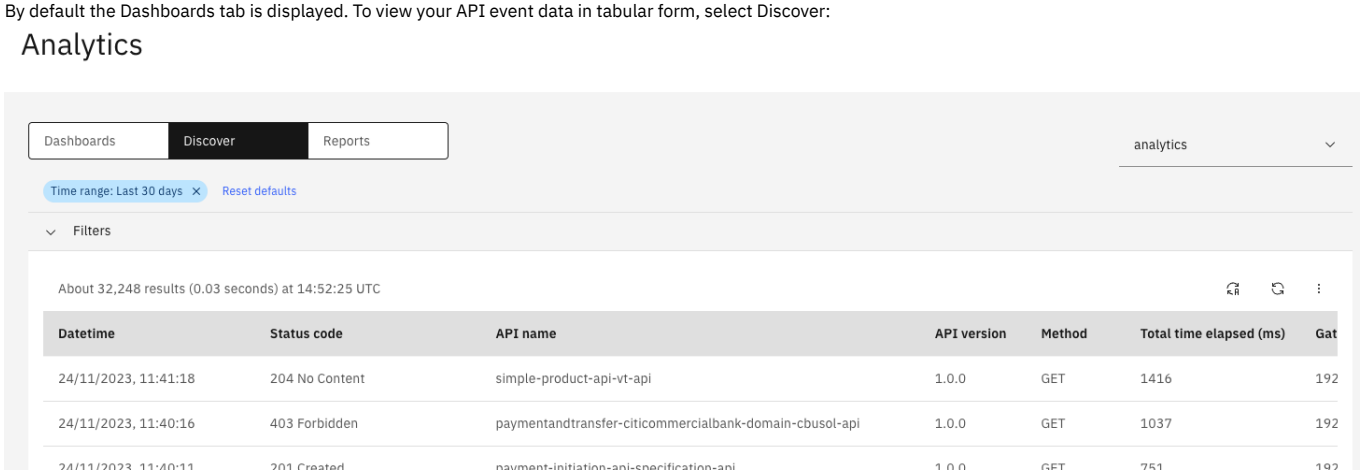
To access the analytics view, click in the navigation panel:



If you have multiple analytics services, use the list to select the analytics service whose API event data you want to view.

Discover view

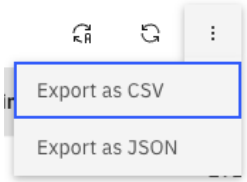
By default the Dashboards tab is displayed. To view your API event data in tabular form, select Discover:



The maximum number of API events that can be accessed from the UI is 10,000. The maximum number of event records that can be displayed per page on the Discover view is 200.

You can customize the columns displayed in the Filters section.

You can export the API event data from this view by clicking Actions at the upper right:



Note: When you export from the Discover view, only the data that is displayed on the current page of results is exported. If you want to export the data from all pages, you must export separately from each page. To export all your analytics data, do not use the UI, see Export your analytics data.
If you see unexpected HTTP OPTIONS events in the Discover view, these events are due to CORS requests from browser-based API test tools. For more information about test tools and the OPTIONS call, see Testing an API.

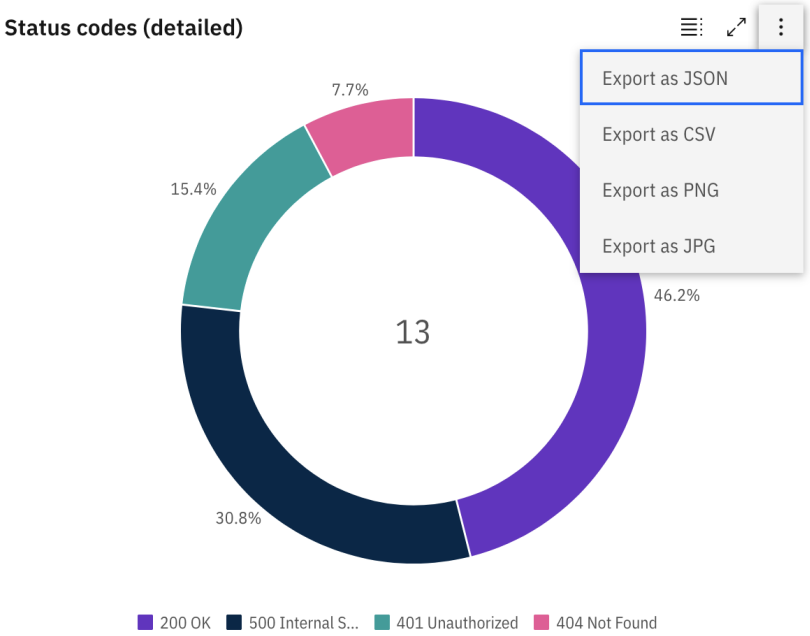
Dashboards view

Eight dashboards are available:

- API Dashboard- contains charts that summarize total API calls, response codes, and response times.
- Product Dashboard - contains charts that show total API calls and application subscriptions per plan.
- Monitoring Latency Dashboard - contains charts that provide response time statistics and data usage.

- Monitoring Status Dashboard - contains charts that show the response codes and success/failure rates of API calls.
- Usage Dashboard - contains charts that show the top 20 APIs, products, and apps.
- Consumption Dashboard - contains charts that show total API consumption in your cloud.
- Gateway Operations Dashboard - contains charts that show API usage and latency by gateway.
- Monitoring Data Dashboard - contains charts that show the APIs, plans, consumer organizations, and applications that send and receive the most data.

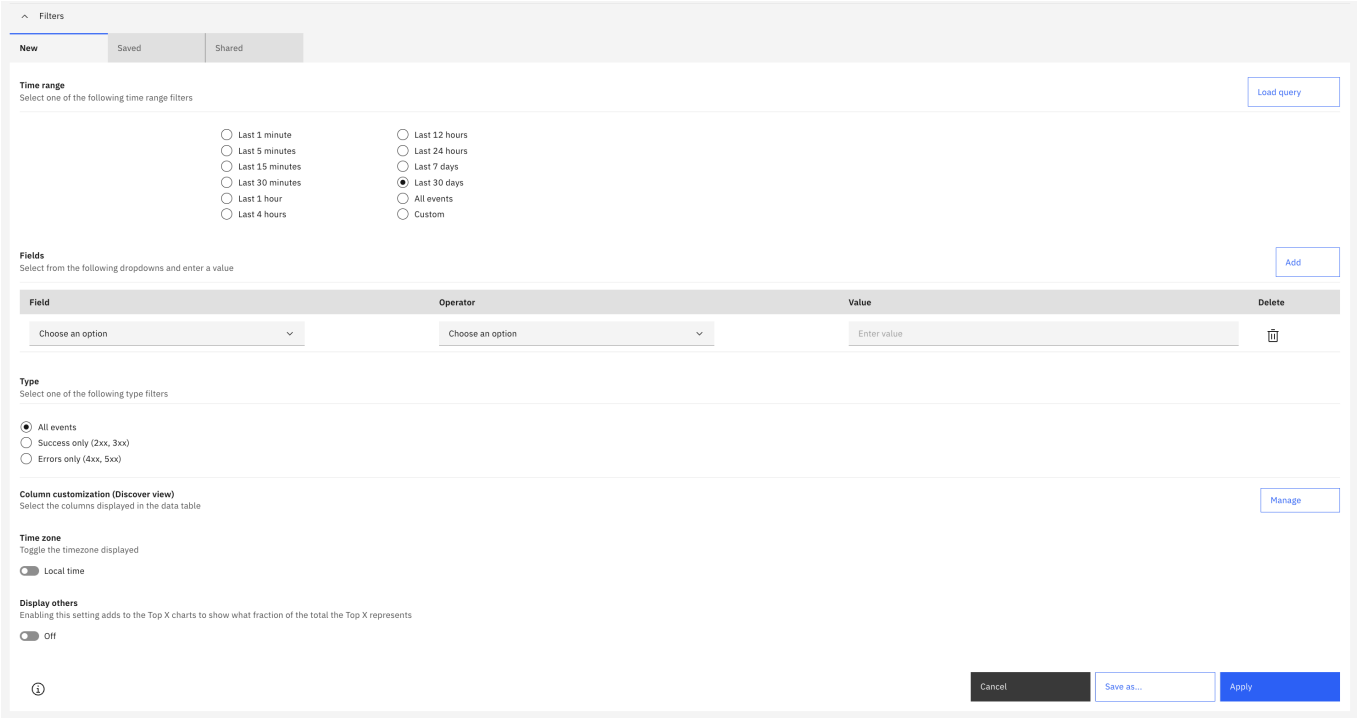
Each chart in a dashboard can be exported as a PNG or JPG image. The source data for each chart can be exported as a JSON or CSV file. To export a chart, click Actions at the upper right:



It is also possible to enlarge the chart to full-screen by clicking . To see the source data in tabular form click .

Filtering displayed data

You can filter the data that is displayed in the Dashboards and Discover view by defining filters (also called "queries"). Expand the Filters section to view, define, and apply queries to your displayed data:

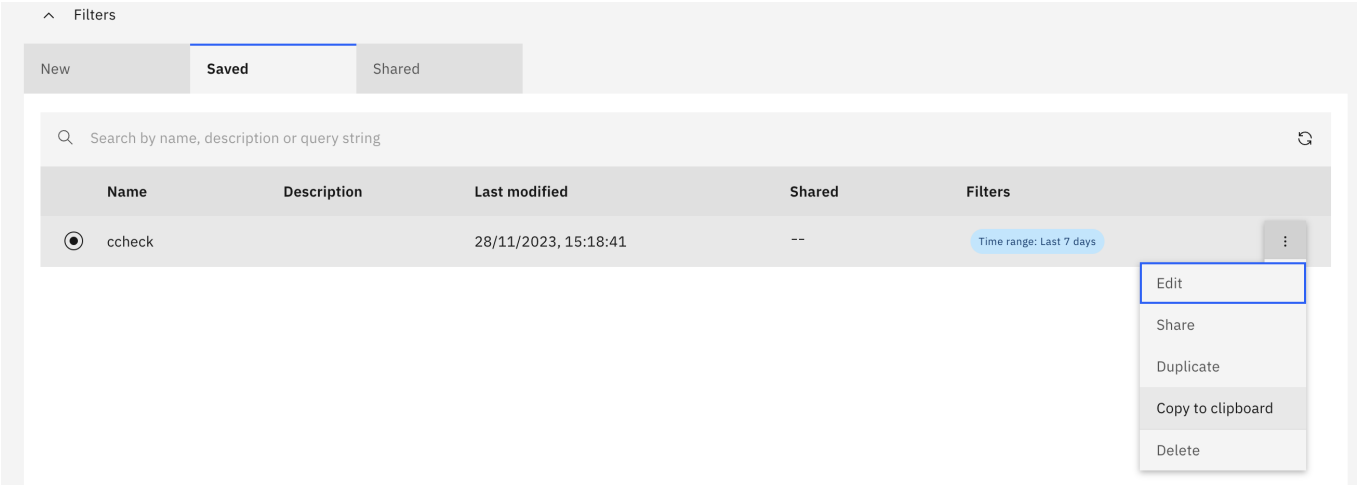


You can save and share the queries that you define:

- Shared queries are visible to all users at the same scope. They are not visible to users in a different scope. For example, a query that Cloud Manager UI users shares is not visible to an API Manager UI user. Within the API Manager UI, a query that is created at catalog scope is not visible to users who are viewing analytics data at

provider organization scope.
- Other users cannot edit or delete queries that you share, but they can create a duplicate query, which they can edit and share.
- All queries have a name and an optional description. The name does not need to be unique.

Saved queries can be exported as strings for use in REST API and toolkit CLI queries, for example:

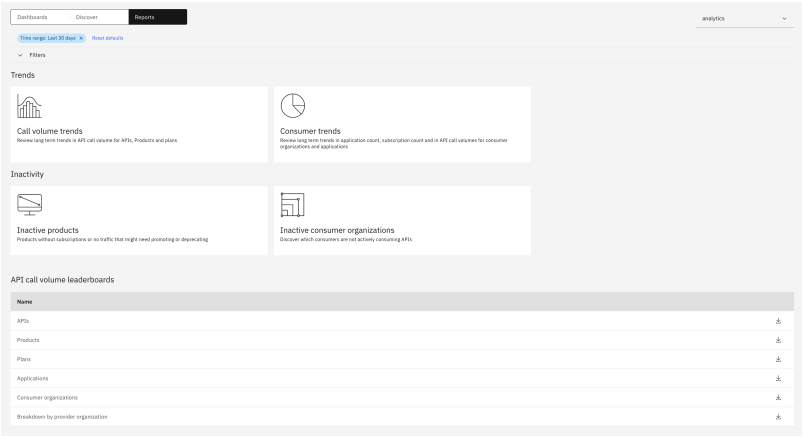


copies the following string to the clipboard:

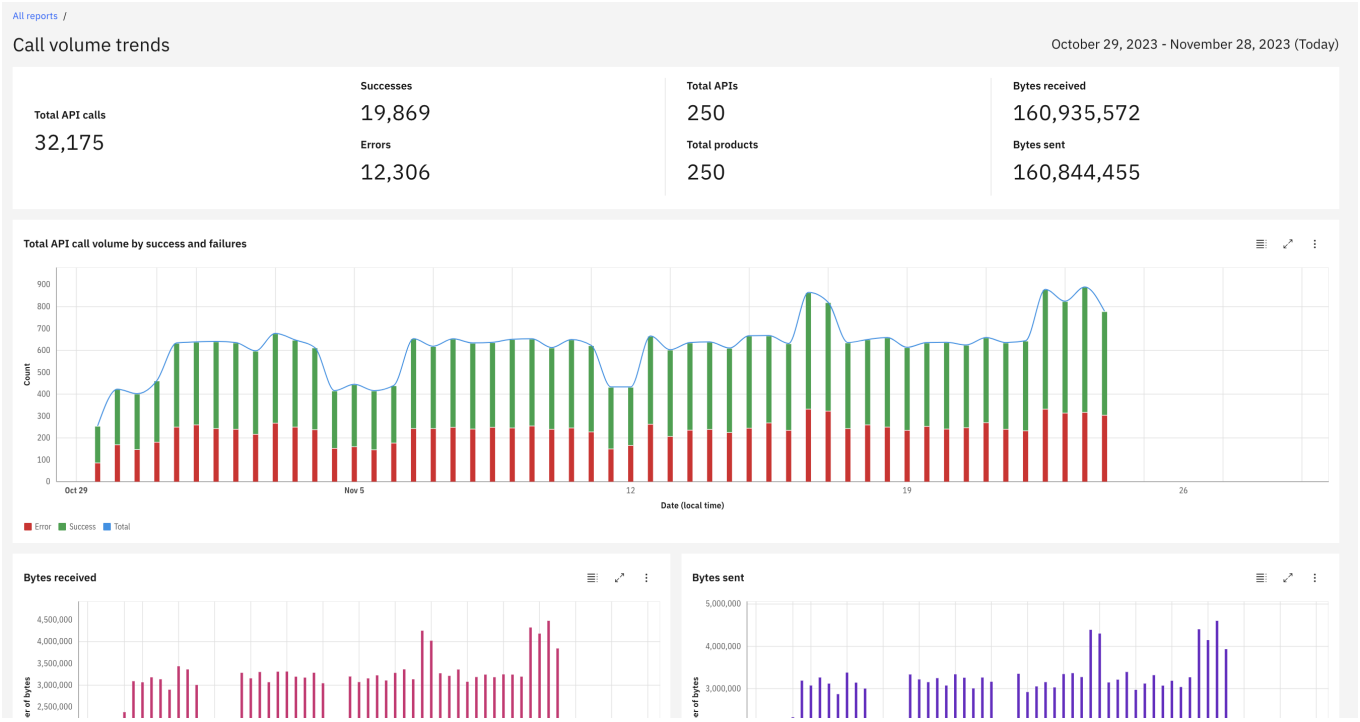
```
timeframe=last7days
```

Reports view

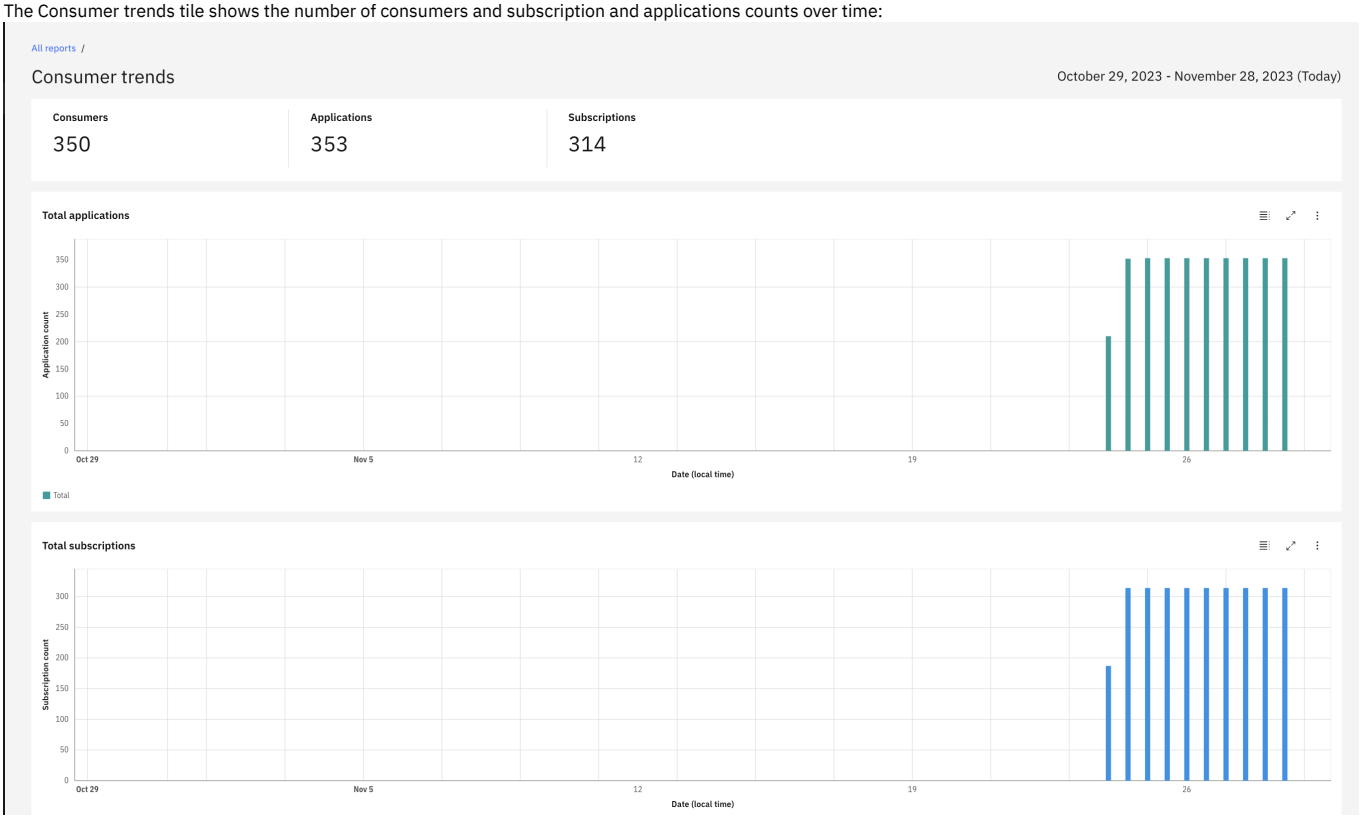
The reports view displays long-term summary data rather than individual API events.
Note: The reports view was introduced in v10.0.6, and does not include data that was generated before upgrade to v10.0.6.
The reports view contains 4 tiles and an API call volume leaderboard



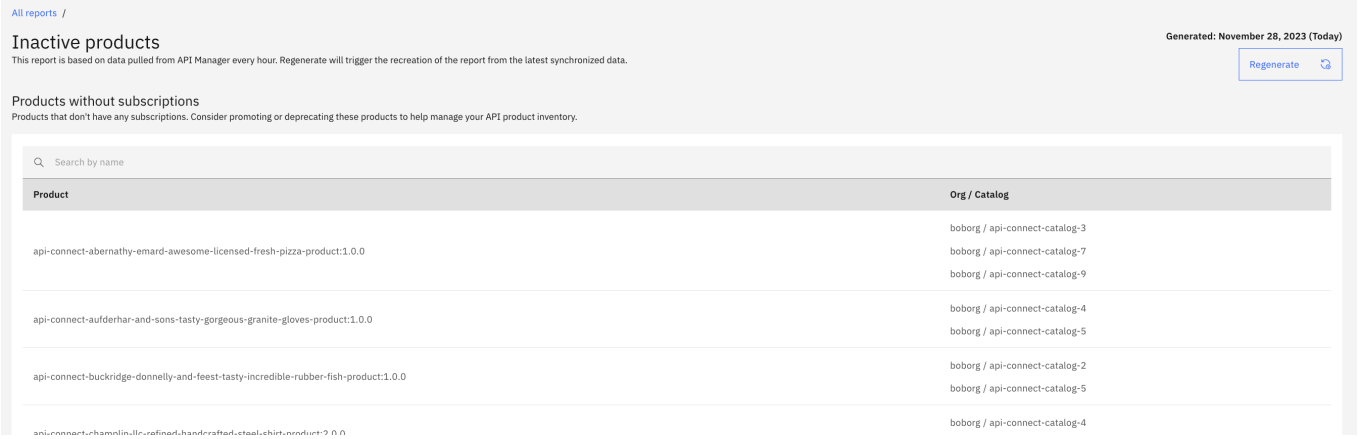
Call volume trends shows your most and least active APIs, total successful and failed API calls, total bytes sent and received, and totals based on scope.



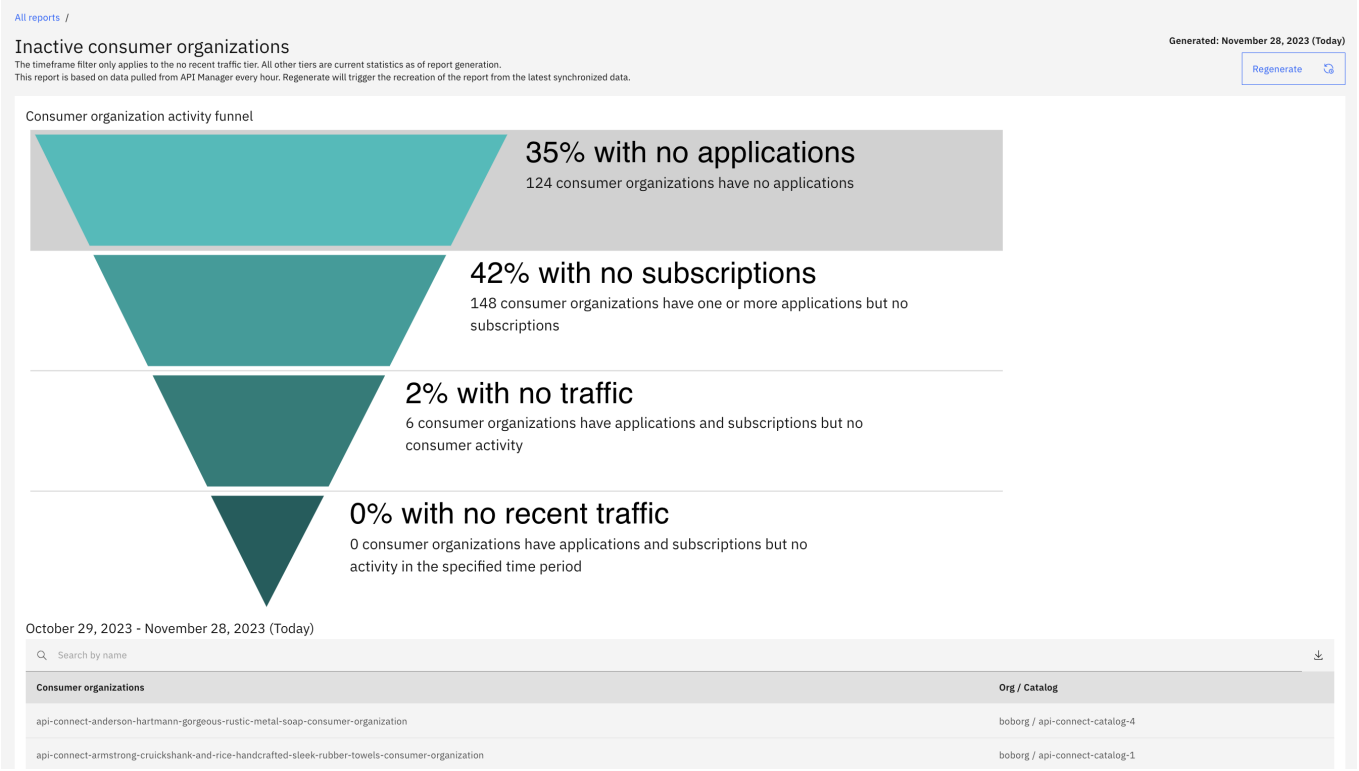
The Consumer trends tile shows the number of consumers and subscription and applications counts over time:



Inactive products shows products that have no subscriptions, and products that have no traffic.



Inactive consumer organizations shows consumer organizations that have no applications, subscriptions, recent traffic, or no traffic at all.



API call volume leaderboards shows leaderboards for APIs, Products, Plans, Applications, Consumer organizations. Click in the name column to see the data for each item.

Accessing analytics in the API Manager UI

How to view, analyze, and export your analytics data from the API Manager UI.

Analytics data is displayed in dashboards in the API Manager UI. Dashboards are scoped at the provider organization, catalog, and space level.

The ability to access analytics data at a catalog or space level depends on your assigned roles and permissions. For more information about catalogs and spaces, see Working with Catalogs and Using syndication in API Connect.

To access the analytics dashboard, complete the following steps:

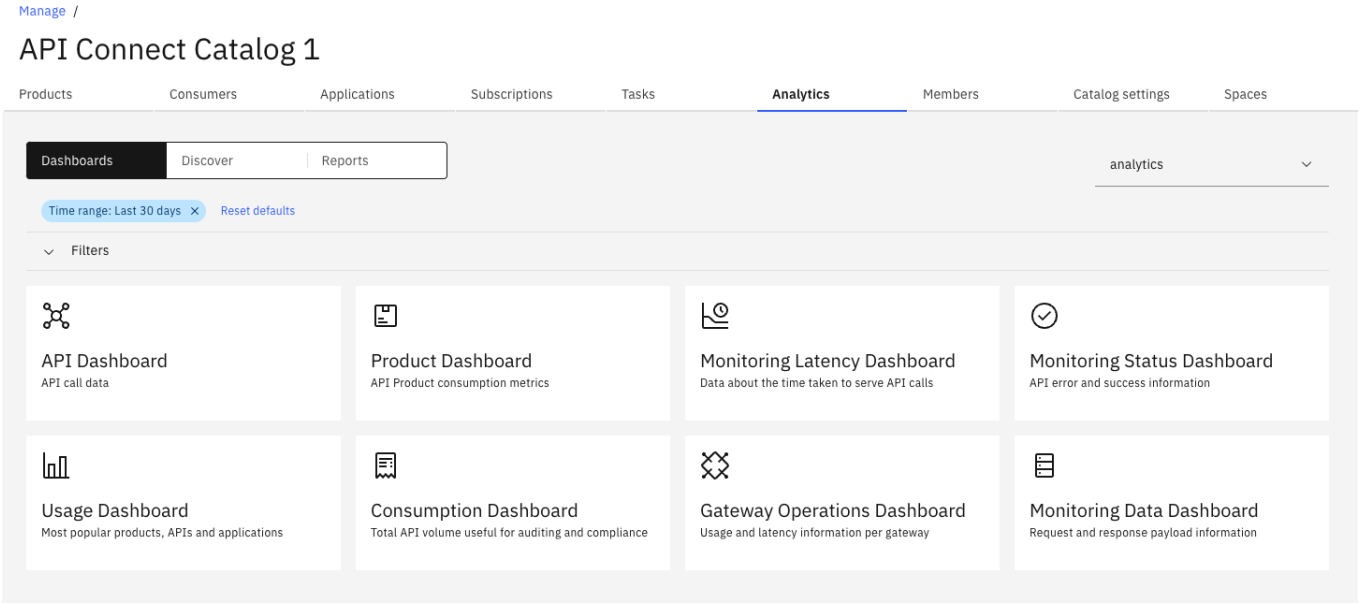
1. In the API Manager UI, select Manage in the navigation panel.
2. From the list of available catalogs in the API Manager Dashboard, select the required catalog.
3. If spaces are enabled and want to see analytics at the scope of a space, then:
    a. Click the Spaces tab.
    b. Select the space that you want to view.
4. Click the Analytics tab.

Analytics dashboards

The analytics dashboards in the API Manager UI show the API event data at the scopes of provider organization, catalog, and space.

The provider organization scoped dashboards are accessed from the analytics icon in the navigation panel.

Catalog scoped dashboards are accessed from the Analytics tab when viewing the catalog:

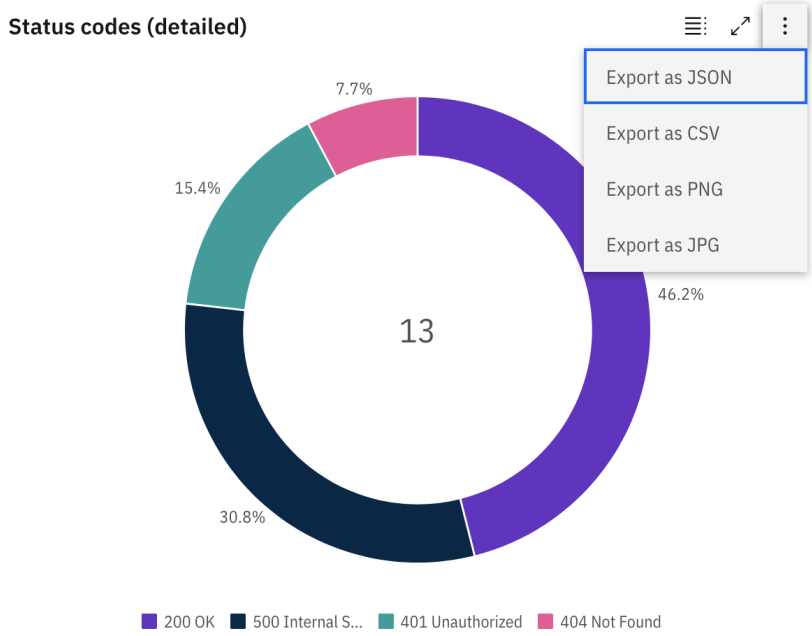


The space scoped dashboards are accessed from the Analytics tab when viewing the space.

Eight dashboards are available:

- API Dashboard- contains charts that summarize total API calls, response codes, and response times.
- Product Dashboard - contains charts that show total API calls and application subscriptions per plan.
- Monitoring Latency Dashboard - contains charts that provide response time statistics and data usage.
- Monitoring Status Dashboard - contains charts that show the response codes and success/failure rates of API calls.
- Usage Dashboard - contains charts that show the top 20 APIs, products, and apps.
- Consumption Dashboard - contains charts that show total API consumption in your cloud.
- Gateway Operations Dashboard - contains charts that show API usage and latency by gateway.
- Monitoring Data Dashboard - contains charts that show the APIs, plans, consumer organizations, and applications that send and receive the most data.

Each chart in a dashboard can be exported as a PNG or JPG image. The source data for each chart can be exported as a JSON or CSV file. To export a chart, click Actions at

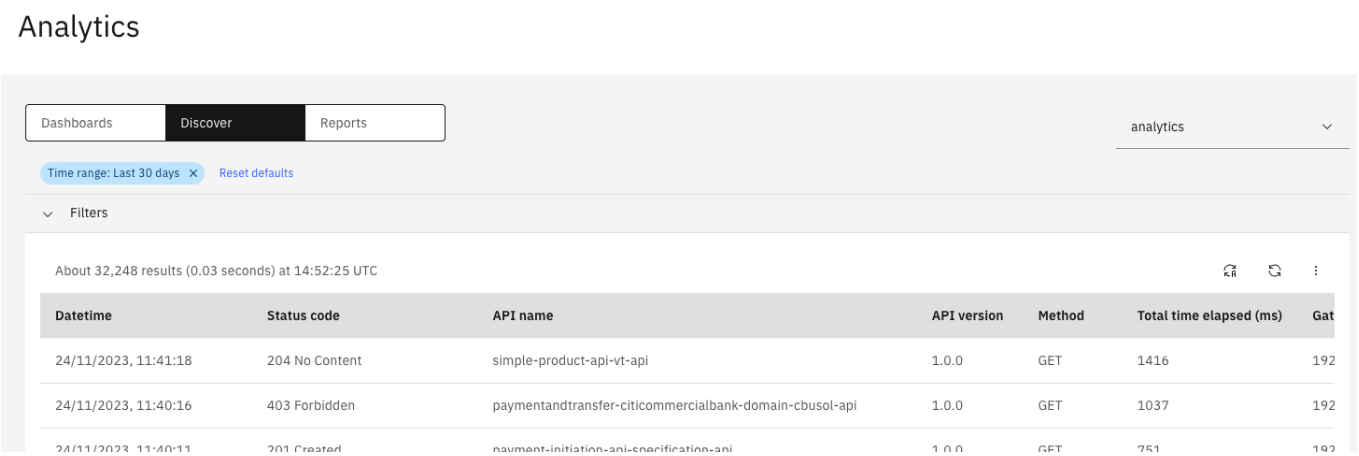


the upper right:

It is also possible to enlarge the chart to full-screen by clicking . To see the source data in tabular form click .

Discover view

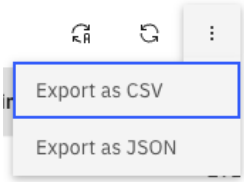
By default the Dashboards tab is displayed. To view your API event data in tabular form, select Discover:



The maximum number of API events that can be accessed from the UI is 10,000. The maximum number of event records that can be displayed per page on the Discover view is 200.

You can customize the columns displayed in the Filters section.

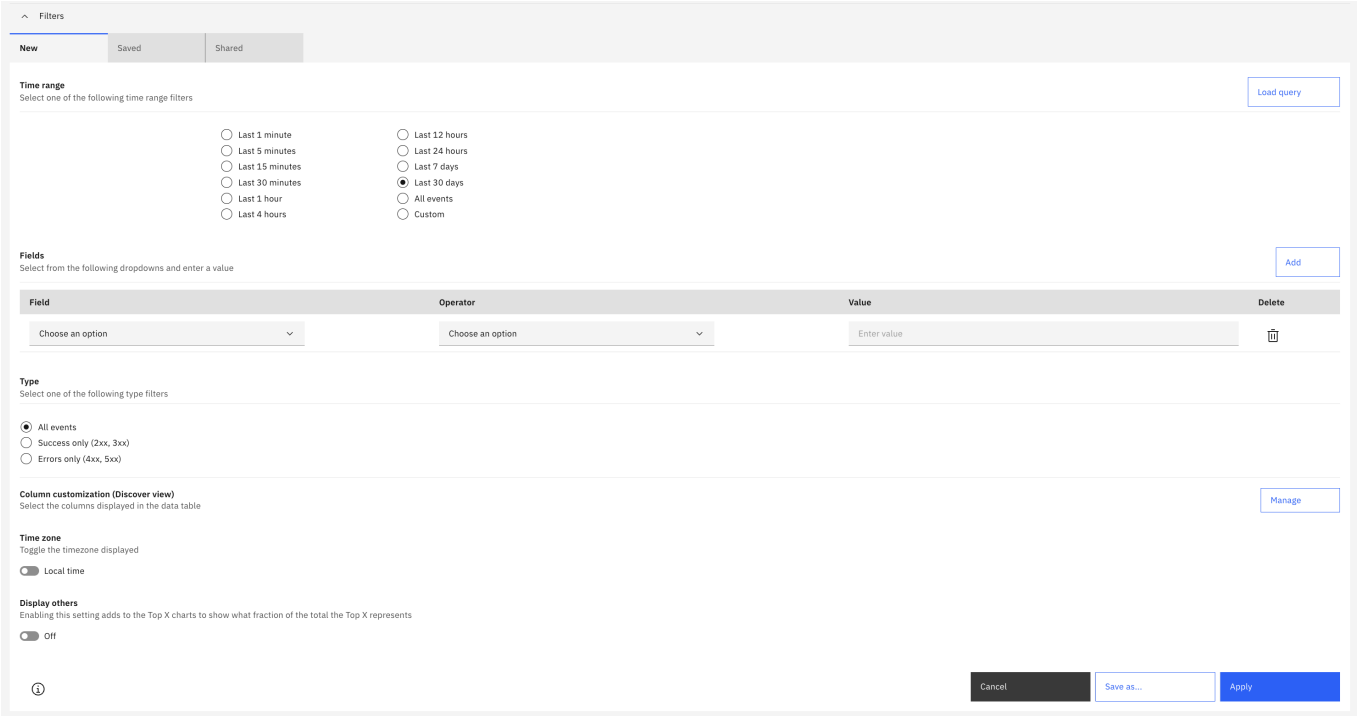
You can export the API event data from this view by clicking Actions at the upper right:



Note: When you export from the Discover view, only the data that is displayed on the current page of results is exported. If you want to export the data from all pages, you must export separately from each page. To export all your analytics data, do not use the UI, see Export your analytics data.
If you see unexpected HTTP OPTIONS events in the Discover view, these events are due to CORS requests from browser-based API test tools. For more information about test tools and the OPTIONS call, see Testing an API.

Filtering displayed data

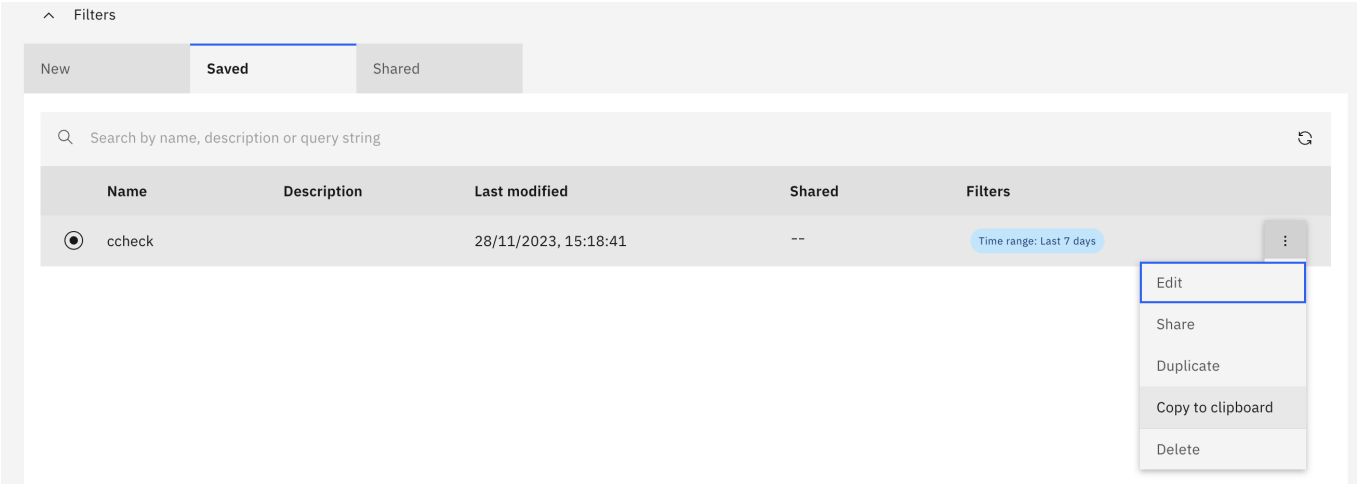
You can filter the data that is displayed in the Dashboards and Discover view by defining filters (also called "queries"). Expand the Filters section to view, define, and apply queries to your displayed data:



You can save and share the queries that you define:

- Shared queries are visible to all users at the same scope. They are not visible to users in a different scope. For example, a query that Cloud Manager UI users shares is not visible to an API Manager UI user. Within the API Manager UI, a query that is created at catalog scope is not visible to users who are viewing analytics data at provider organization scope.
- Other users cannot edit or delete queries that you share, but they can create a duplicate query, which they can edit and share.
- All queries have a name and an optional description. The name does not need to be unique.

Saved queries can be exported as strings for use in REST API and toolkit CLI queries, for example:

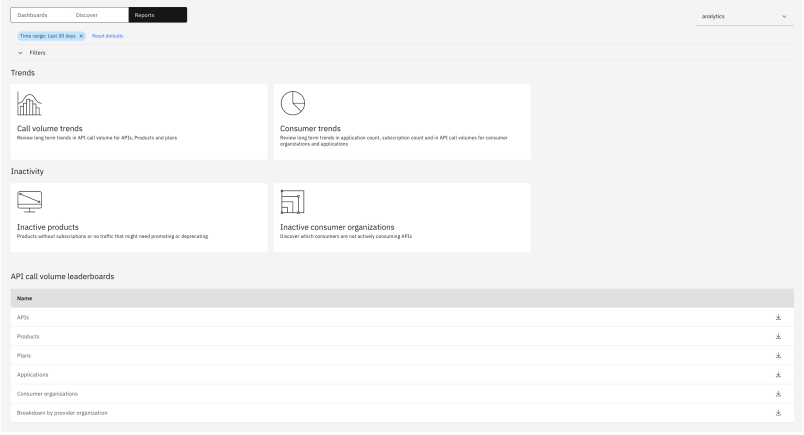


copies the following string to the clipboard:

```
timeframe=last7days
```

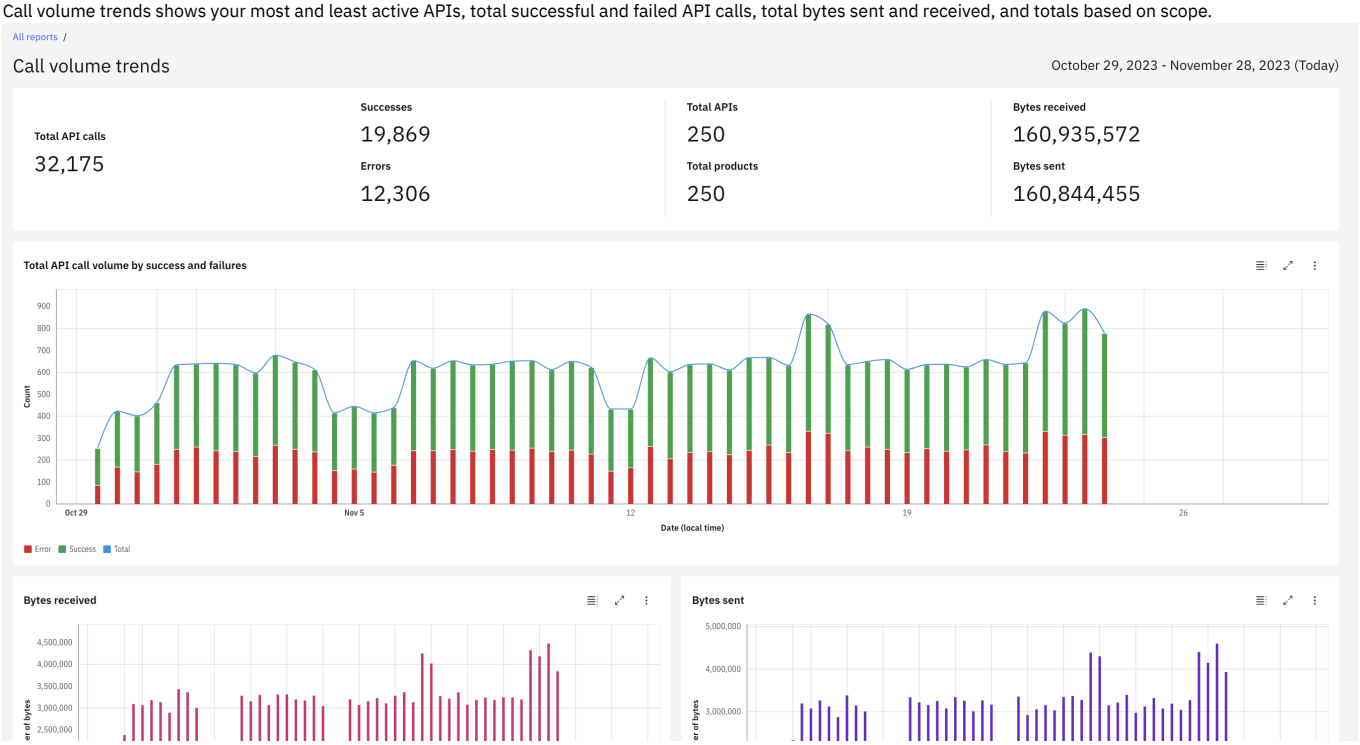
Reports view

The reports view displays long-term summary data rather than individual API events.

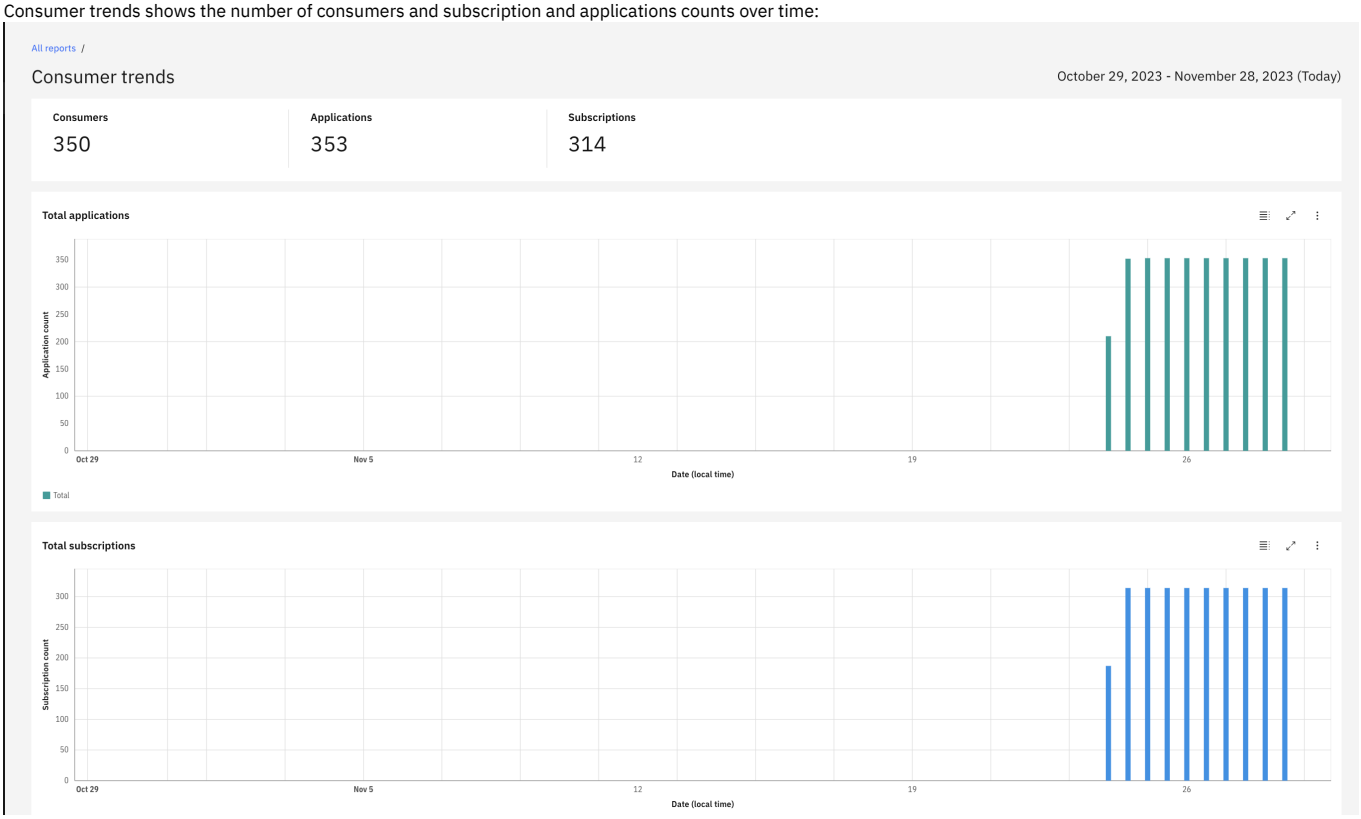


Note: The reports view was introduced in v10.0.6, and does not include data that was generated before upgrade to v10.0.6.

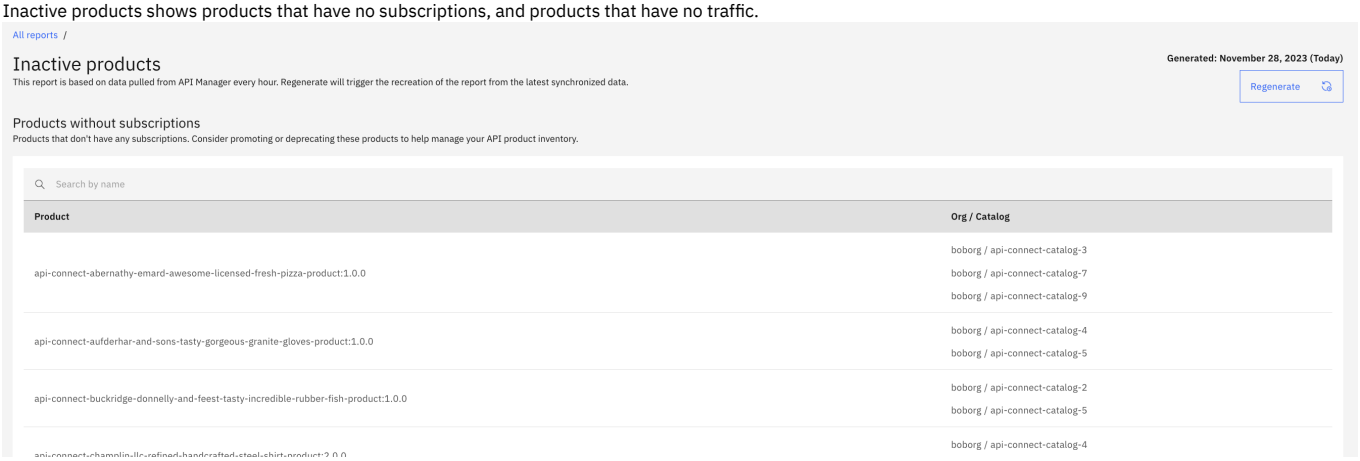
Call volume trends shows your most and least active APIs, total successful and failed API calls, total bytes sent and received, and totals based on scope.



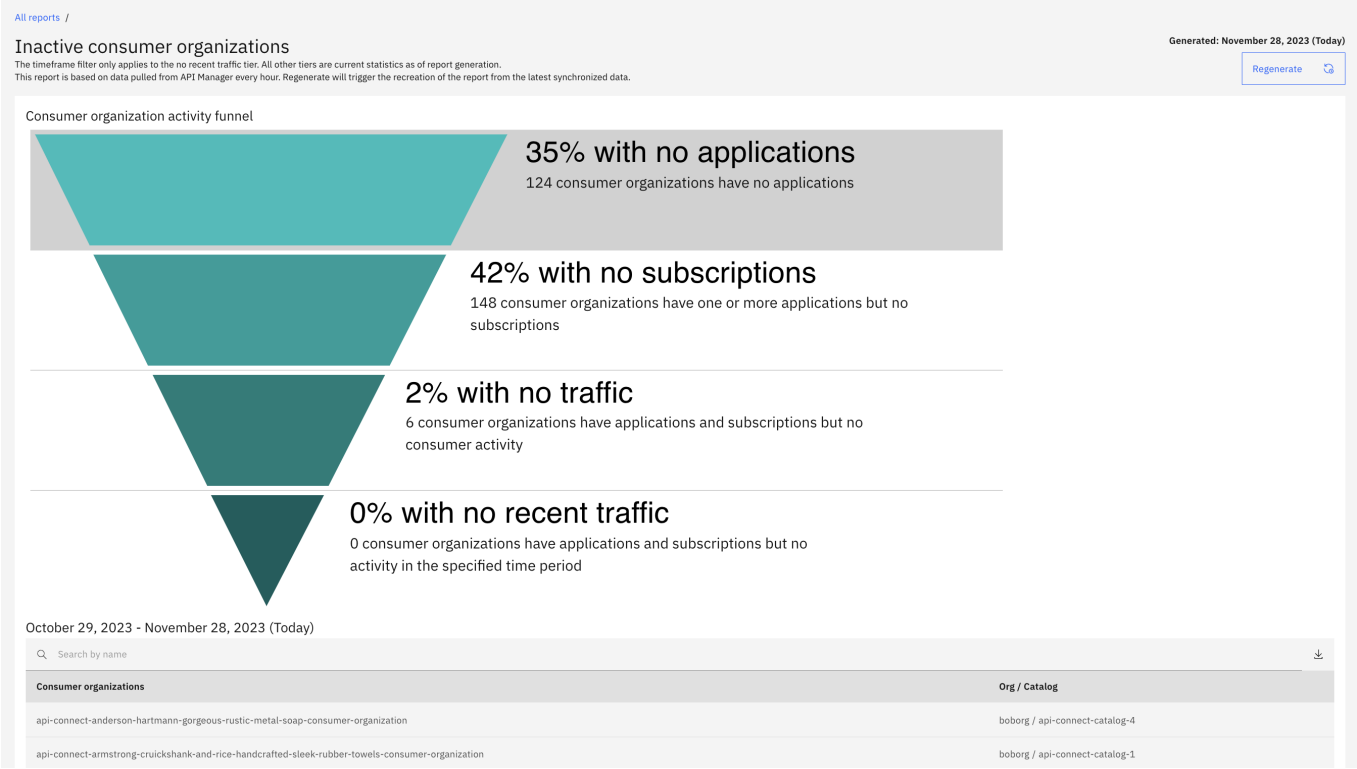
Consumer trends shows the number of consumers and subscription and applications counts over time:



Inactive products shows products that have no subscriptions, and products that have no traffic.



Inactive consumer organizations shows consumer organizations that have no applications, subscriptions, recent traffic, or no traffic at all.



The API call volume leaderboards section shows leaderboards for APIs, Products, Plans, Applications, Consumer organizations. Click in the name column to see the data for each item.

Related concepts

- Catalogs, spaces, and analytics

Accessing analytics in the Developer Portal

How to view your analytics data from the Developer Portal.

Restriction: The Developer Portal sites show analytics data from the analytics service that is associated with the gateways that are used by the catalog. If a Developer Portal site's catalog uses more than one gateway service, then those gateway services must be associated with the **same** analytics service. If multiple analytics services are used by a catalog's gateways, then no analytics data is shown in the portal site corresponding to that catalog. Analytics services are associated with gateway services in the Cloud Manager UI, see Associating an analytics service with a gateway service.

Viewing organization scoped analytics data

1. Click the arrow next to your consumer organization name on the Developer Portal home page, and click My organization.
2. Click the Analytics tab.

From the Analytics tab, you can see your organization analytics, which include: API stats, total calls, and total errors.

Viewing application scoped analytics data

1. Click Apps from the Developer Portal dashboard.
2. Click the name of the application for which you want to view analytics data.

From the Dashboard tab, you can see your application analytics, which include: API stats, total calls, and total errors.

Counting total API calls for your analytics service

Count the API calls for the analytics subsystem and see the total categorized by response status.

Note: If your API Connect bases costs on usage, you can use the following methods to track this information so that you can accurately report it. If your organization did not deploy the analytics subsystem, then this feature is not available. If you deployed analytics but offload your data and disabled local storage, then you must capture usage information from the third-party system where you offload data.
You can see the total API calls over time for each analytics service in the consumption dashboard. To see the consumption dashboard, navigate to the Cloud Manager UI analytics view: Accessing analytics in the Cloud Manager UI.

Alternatively, the Analytics CLI can be used to get your API count:

1. Login as the cloud admin user:

   ```
   apic login --server <platform api endpoint> --username admin --realm admin/default-idp-1
   ```

2. Use the following CLI command to retrieve a count of all analytics events recorded by the analytics service:

   ```
   apic -m analytics events:cloudCount --server <platform api endpoint> --analytics-service <analytics service name> --format
   json
   ```

   where *<analytics service name>* is the name of analytics service as configured in the Cloud Manager UI.
3. If required, additional filters can be applied to the results. For example, `--timeframe last12hours`. See the `help` output from: `apic -m analytics events:cloudCount -h` for more details.

Export your analytics data

Export your analytics data with either the toolkit CLI or the REST API.

The analytics export function in the UIs export just the data that is displayed in the UI, and is limited to 200 API events per page.

To export up to 10,000 API event records, use the toolkit CLI.

To export all your analytics data, use the REST API.
Note: If you plan to export you API event data regularly, consider using the offload feature instead. See Offload to third-party systems.

Exporting with the toolkit CLI

1. Follow the steps to install and configure the toolkit CLI.
2. Login with the correct user for the scope of analytics data you want to export. For example, to export all API event from a provider organization called `prod`, login as a member of that organization:

   ```
   apic login --server <platform api endpoint> --username <provider org member> --realm provider/default-idp-2
   ```

3. Use the `events` command to export your analytics data. For example, to export the events for a provider organization:

   ```
   apic -m analytics events:orgList --server <platform api endpoint> --analytics-service <analytics service name> --org <org
   name> --format=json
   {
       "total": 200,
       "search_time": 7,
       "events": [
           {
               "log_policy": "activity",
   ...
   ```

   Note: Data is returned as a JSON array, each element is a single API event record.

Exporting with the REST API

1. Follow the steps to get started with the Analytics REST API.
2. Use the `events` API to export your data. For example, to get all analytics API event records from a provider organization, specify the organization's API path:

   ```
   curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url
   'https://<management_server_api_endpoint>/analytics/<analytics_service>/orgs/<provider_organization>/events'

   {
       "total": 45543,
       "search_time": 36,
       "events": [...]
   ```

To get just the events for a catalog, specify the catalog in the API path; for example:

`/analytics/<analytics_service>/catalogs/<provider_organization>/<catalog>/events`

Allowing API Connect cloud admin users to view provider organization analytics data

The owner of a provider organization can enable or disable the sharing of analytics data with API Connect cloud administrators.

About this task

In previous releases of API Connect, all analytics data from provider organizations was available to cloud admin users by default. Beginning with version 10.0.7.0, the data is not shared with cloud admins by default; the owner of a provider organization must enable the sharing of analytics data with cloud admins.

This setting does not apply to total API call volume data. Cloud admins can always see the total volume of API calls across all provider organizations and for specific gateways. The setting applies only to analytics data that identifies the provider organization.

Procedure

To enable sharing of analytics data with cloud administrators, complete the following steps.

1. In API Manager, click Settings > Analytics > Share analytics data.
2. Click the slider to set it to On.

Analytics CLI

How to access your analytics data with the toolkit CLI.

About this task

The toolkit CLI provides commands for viewing your analytics data, and also for configuring analytics storage settings. To view organization scoped event data, login to the toolkit CLI with provider organization user credentials. To view cloud scoped event data and analytics storage settings, login to the toolkit CLI as the cloud admin user.

Procedure

1. Install the toolkit: Setting up the API Connect toolkit
2. Log in with the CLI.
   For example:

   ```
   apic login --server <platform api endpoint> --username <username> --realm <realm>
   ```

   Where:
   - `<platform api endpoint>` is the FQDN or base URL of your platform API endpoint, for example: `platform.example.com` or `https://platform.example.com`
   - `<username>` is the provider org owner. To access cloud scoped data, login as the admin user.
   - `<realm>` is the user registry realm for the specified `<username>`. By default the realm for provider organization owners is `provider/default-idp-2`. The realm for cloud admin users is `admin/default-idp-1`.

   For more information, see: Toolkit CLI Login.
3. View the available toolkit analytics commands by using `-m analytics -h`. Example:

   ```
   apic -m analytics -h
   ...
   Available Commands:

     Creating applications and artifacts
        config                                  Manage configuration variables

     Publishing to the cloud
        clustermgmt                             Clustermgmt operations
        dashboards                              Dashboards operations
        events                                  Events operations
        filterparams                            Filterparams operations

     Other commands
        licenses                                Review the license for API Connect
        validate                                Validate an API or product definition
        version                                 Get the APIConnect toolkit version
   ...
   ```

4. View your analytics events by using the `events` commands:

   ```
   apic -m analytics events -h
   Type "apic [command] --help" for help on the following related commands:
         events:catalogCount  Count events at catalog scope.
         events:catalogGet    Get specific event at catalog scope.
         events:catalogList   List events at catalog scope.
         events:cloudCount    Count events at cloud scope.
         events:cloudGet      Get specific event at cloud scope.
         events:cloudList     List events at cloud scope.
         events:orgCount      Count events at organization scope.
         events:orgGet        Get specific event at organization scope.
         events:orgList       List events at organization scope.
         events:spaceCount    Count events at space scope.
         events:spaceGet      Get specific event at space scope.
         events:spaceList     List events at space scope.

   # To get a count of all API events at cloud level:
   apic -m analytics events:cloudCount --analytics-service <analytics service name> --server <management server api endpoint>
   ```

```
--format=json
{
    "total": 200
}

# To get all the cloud level events:
apic -m analytics events:cloudList --analytics-service <analytics service name> --server <management server api endpoint>
--format=json
{
    "total": 200,
    "search_time": 7,
    "events": [
        {
            "log_policy": "activity",
...
```

Analytics REST API

How to access your analytics data with the REST API.

Before you begin

API Connect REST API calls are authenticated by using a bearer token in the authorization header. Use the toolkit credentials to request a bearer token. Toolkit credentials can be found in the API Manager UI:

1. Log in to the API Manager UI.
2. From the home page, click the Download toolkit tile.
3. Download the Toolkit credentials.
4. Open the downloaded `credentials.json` file:

```
...
  "toolkit": {
    "endpoint": "https://mgmt.api.example.com/api",
    "client_id": "7409693f-f726-48b4-8909-7c0d26f13e81",
    "client_secret": "5feeb0be-17f8-41a4-96d6-d40f33d69ef6"
  },
...
```

Take note of the `toolkit.client_id`, `toolkit.client_secret`, and `toolkit.endpoint` from this file. You use these values to get your bearer token.

Procedure

1. Use your credentials to request a bearer token:

```
curl -v -k -X POST -d '{"username": "<username>", "password": "<password>", "realm": "<realm>", "client_id": "
<client_id>", "client_secret": "<client_secret>", "grant_type": "password"}' -H 'Content-Type: application/json' -H
'Accept: application/json' https://<platform api endpoint>/token
```

   Where:
   - *<username>* is the provider org owner. To access cloud scoped data, login as the admin user.
   - *<password>* is the password for the specified *<username>*.
   - *<realm>* is the user registry realm for the specified *<username>*. By default the realm for provider organization owners is `provider/default-idp-2`. The realm for cloud admin users is `admin/default-idp-1`.
   - *<client_id>* is the client ID from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
   - *<client_secret>* is the client secret from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator.
   - *<platform api endpoint>* is the platform API endpoint from the "toolkit" section of the `credentials.json` file, or as provided by your API Connect cloud administrator. It might have the path "api" appended to it, so full string is: `https://<platform api endpoint>/api/token`

   The bearer token is returned in the access_token property:

```
{
    "access_token": "<bearer_token>",
    "token_type": "Bearer",
    "expires_in": 28800
}
```

   Note the `expires_in` value, which is the number of seconds before the token expires. After expiry, you must request a new bearer token.

2. Use the returned bearer token to call the analytics REST API:

```
# This call to /orgs/<provider_organization>/events requires a bearer token that was requested with provider org
credentials:
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url 'https://<platform api
endpoint>/analytics/<analytics_service>/orgs/<provider_organization>/events'

{
    "total": 300,
    "search_time": 3,
    "events": [...]
}

# This call to /cloud/events requires a bearer token that was requested with admin credentials:
curl -v -k -H 'Accept: application/json' -H 'Authorization: Bearer <bearer_token>' -X GET --url 'https://<platform api
endpoint>/analytics/<analytics_service>/cloud/events'

{
    "total": 45543,
```

```
    "search_time": 36,
    "events": [...]
```

Where:

- ***<platform api endpoint>*** is the **toolkit.endpoint**, but with the **/api** at the end replaced with **/analytics** to access the analytics APIs.
- ***<analytics_service>*** is the name of the analytics service.
- *<bearer_token>* is the token from step 1. Do not wrap the token with quotation marks.

What to do next

Complete documentation of the analytics REST API is here: Analytics REST API.

Troubleshooting analytics

How to diagnose and resolve common issues with the analytics subsystem.

- **No analytics data in UIs**
  How to check that your analytics subsystem is receiving API event data from the gateways.
- **Running out of storage space**
  Actions that you can take if your analytics data is using more persistent storage space than you expected.
- **Offload failure**
  What to check if offload to a third-party system is failing.
- **Gathering postmortem logs**
  The logs to gather for upload to a support case.

No analytics data in UIs

How to check that your analytics subsystem is receiving API event data from the gateways.

Note: **VMware users:** The operations to confirm API event data is arriving from the gateways must be run from inside the analytics VM. To access your analytics VM:

1. Use an SSH client to login as the **apicadm** user:

   ```
   ssh apicadm@<analytics VM FQDN>
   ```

2. **sudo** to root:

   ```
   sudo -i
   ```

The **kubectl** commands in these operations specify a *<namespace>* argument. On VMware, replace *<namespace>* with "default". Alternatively you can omit **-n *<namespace>*** from the commands completely.

1. The **mtls-gw** is the pod that receives the API events from the gateway. Identify the name of your **mtls-gw** pod or pods with:

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

   Expected output:

   ```
   NAME                                                              READY   STATUS      RESTARTS   AGE
   ...
   analyt-mtls-gw-7665f5d445-4cx4n                                     1/1     Running     0          3d19h
   ```

2. For each **mtls-gw** pod, run the following command to see the logs of that pod:

   ```
   kubectl logs <mtls-gw pod name> -n <namespace>
   ```

3. The gateway sends API event data to the **mtls-gw** pod as an HTTP POST call to the path **/x2020/v1/apievent/_bulk**.
   Every time that the **mtls-gw** pod receives API event data from the gateway, it logs the following message, known as a **_bulk** message:

   ```
   10.254.6.1 - - [22/Sep/2021:18:05:54 +0000] "POST /x2020/v1/apievent/_bulk HTTP/1.1" 200 2 "-""-" 2841 0.072 0.070
   ```

   Check the **mtls-gw** pod logs for recent **_bulk** messages.

   If you find no **_bulk** messages in the **mtls-gw** logs, then the analytics subsystem is not receiving API event data from the gateway. Check the network connection between your gateway and analytics subsystems, and review *Analytics endpoints* in the gateway documentation.

   The HTTP return code is shown in the **mtls-gw** log messages. In the example, this code is 200, which means the API event data was received successfully by the analytics subsystem. Whatever the return code is, if your **mtls-gw** pod logs contain recent "_bulk" messages and you do not see any analytics data for this time period in the UI, then gather postmortem logs and open an IBM support case.

Running out of storage space

Actions that you can take if your analytics data is using more persistent storage space than you expected.

Reduce retention period

The simplest remedy to running out of storage space is to reduce the data retention period.

By default, API event records are retained for 30 days on a one replica deployment, and 90 days on a three replica deployment.

If your analytics subsystem was deployed more recently than your retention period (for example, less that 30 days ago for a default one replica deployment), and you are running out of storage space, then there is a risk that your persistent storage fills up before the retention period is reached. Check in the Cloud Manager UI how many days back your API event data goes, and set the retention period to a lower number. For example, if you see API event records from 15 days ago, reduce your storage requirement by one-third by setting your retention period to 10 days. For more information about data retention, see Retention and rollover.

VMware users: add extra disk

You can increase the space available for your analytics data by adding a disk, see Adding disk space to a VMware appliance.
Warning: Do not attempt to increase the size of the existing data disk. Changing the disk size corrupts your existing analytics data.

Kubernetes and Openshift users: Reinstall with larger PVCs

It is not possible to increase PVC size after installation. If you require more space for your analytics data, the analytics subsystem must be reinstalled. Follow these steps:

1. Take a backup of your analytics subsystem: Kubernetes, OpenShift, and Cloud Pak for Integration.
2. Reinstall your analytics subsystem with a larger PVC size:
   - Kubernetes
   - OpenShift and Cloud Pak for Integration
3. Restore your analytics data from your backup: Recovering from a disaster.

Reduce space used by new API event data

To reduce the storage required for new analytics data, update the activity log settings of your APIs: Activity log policy.
Tip: If you're not able to update the activity log setting for your APIs, you can disable response payload logging for all APIs with ingestion filters. For example:

```
filter {
  if "apicapievent" in [tags] {
     mutate {
           remove_field => ["response_body"]
        }
  }
}
```

What to do if your storage is 100% full

If your storage space is 100% full, your analytics subsystem might be unresponsive, so reducing the retention period is not possible. In this scenario, follow the disaster recovery steps: Recovering from a disaster.

Offload failure

What to check if offload to a third-party system is failing.

- Check the ingestion pod logs for error messages that are related to offload:
  - Kubernetes, OpenShift, and Cloud Pak for Integration:

    ```
    kubectl -n <namespace> logs <analytics ingestion pod name>
    ```

  - VMware: Find the ingestion pod logs in Gathering logs for a VMware environment.

  A common problem is TLS handshaking errors, such as untrusted or expired certificates.
- Check logs on your third-party system.
- Verify that the specified URL or the host and port values for the target system are accurate.
- Verify that the server or cluster of servers in the target system are running and reachable. If applicable, also verify that firewall settings are not causing connection problems.
- If offloading data to an Elasticsearch or Kafka target, verify that your configuration settings are valid; for example, the user credentials for authenticating to Elasticsearch or the Kafka topic name.

Gathering postmortem logs

The logs to gather for upload to a support case.

The generate_postmortem.sh script

The generate_postmortem.sh script when run with the `--diagnostic-analytics` argument gathers all the information that is typically needed for an analytics problem. The script output includes:

- The analytics pod logs.
- YAML files that describe all the analytics subsystem Kubernetes artifacts.
- The OpenSearch cluster status.

Use the generate_postmortem.sh script, as described for your platform:

- Gathering logs for a VMware environment
- Gathering logs on Kubernetes and OpenShift

Ensure that you use the `--diagnostic-analytics` argument when you run generate_postmortem.sh.

Gathering the pod logs

If the generate_postmortem.sh script fails, the pod logs can be gathered manually:

- **VMware:** SSH into your analytics VM and run **apic logs**:
  1. Use an SSH client to login as the `apicadm` user:

     ```
     ssh apicadm@<analytics VM FQDN>
     ```

  2. `sudo` to root:

     ```
     sudo -i
     ```

  3. Run the **apic logs** command:

     ```
     apic logs
     ```

  4. Two compressed files are created: appliance-logs, and node-logs. Copy both files off the VM and upload to the support case.
- **Kubernetes and Openshift:** Get the logs from all analytics pods in your environment. Use the command:

  ```
  kubectl -n <namespace> logs <analytics pod name> > <analytics pod name>.log
  ```

  where:
  - *<namespace>* is the namespace of your analytics subsystem.
  - *<analytics pod name>* is the name of the analytics pod. See the list of analytics pods.

  The logs for each pod are written to the file *<analytics pod name>*.log. Upload all the generated logs to the support case.

Getting the OpenSearch cluster status

If the generate_postmortem.sh script fails you can use the toolkit CLI to get the status of your OpenSearch cluster, and a list of the indices.

1. Follow the steps to install and configure the toolkit CLI.
2. Login to the CLI as the cloud admin user:

   ```
   apic login --server <platform api endpoint> --username admin --realm provider/default-idp-1
   ```

3. Get the OpenSearch cluster status with:

   ```
   apic --mode analytics clustermgmt:catIndices --server <platform api endpoint> --analytics-service <analytics service name>
   --format json
   ```

4. Get the details of the OpenSearch indices with:

   ```
   apic --mode analytics clustermgmt:catIndices --server <platform api endpoint> --analytics-service <analytics service name>
   --format json
   ```

Upload the cluster status and indices output to the support case.

Reference information

Reference information for IBM® API Connect, including summary reference material for the developer toolkit command-line tool (CLI), a link to the API Connect Version 10.x Whitepaper, and information about the API Connect platform REST APIs.

Reference item	Description
Command-line tool reference for the developer toolkit	Summary reference material for each of the available commands for the developer toolkit command-line tool (CLI). (For more detailed information about how to use the command-line tool, see Using the developer toolkit command-line tool.)
Command-line tool reference for the Developer Portal	Summary reference material for each of the available commands for the Developer Portal command-line tool (CLI). (For more detailed information about how to use the command-line tool, see Using the Developer Portal command-line tool.)
API Connect REST APIs	The platform REST APIs that can be used to manage the API Connect cloud configuration, access the API provider capability, and use the consumer API microservice.
API Connect v10.x Whitepaper	A comprehensive technical guide to best practices, considerations, and deployment options for API Connect. The whitepaper covers the major components of API Connect, as well as considerations for configuring different clouds and environments, to help users with their API strategies. The target audience for the whitepaper are solution and integration architects.

apic

APIConnect toolkit 8796fc7789531c914ecbd402c6ff51ca96248b06 (Built 2023-12-05T21:51:07Z)

Synopsis

APIConnect toolkit 8796fc7789531c914ecbd402c6ff51ca96248b06 (Built 2023-12-05T21:51:07Z)

```
apic [flags]
```

Options

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
  -h, --help                   Help for apic
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic activations

Activations operations

Synopsis

Activations operations

```
apic activations [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for activations
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic activations:clear

Activations clear operations

Synopsis

Activations clear operations

```
apic activations:clear [flags]
```

Possible Usages

```
    apic activations:clear [flags] --scope org
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for activations:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic activations:delete

Activations delete operations

Synopsis

Activations delete operations

```
apic activations:delete [flags]
```

Possible Usages

```
apic activations:delete [flags] --scope org ACTIVATION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for activations:delete
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic activations:get

Activations get operations

Synopsis

Activations get operations

```
apic activations:get [flags]
```

Possible Usages

```
apic activations:get [flags] --scope org ACTIVATION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for activations:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic activations:list

Activations list operations

Synopsis

Activations list operations

```
apic activations:list [flags]
```

Possible Usages

```
      apic activations:list [flags] --scope org
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for activations:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic analytics

Analytics operations

Synopsis

Analytics operations

```
apic analytics [flags]
```

Options

```
  -h, --help   Help for analytics
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic analytics:create

Analytics create operations

Synopsis

Analytics create operations

```
apic analytics:create [flags]
```

Possible Usages

```
      apic analytics:create [flags] --scope space ANALYTICS_FILE
      apic analytics:create [flags] --scope catalog ANALYTICS_FILE
```

Options

```
      --analytics-service string   Analytics Service name or id (required)
  -c, --catalog string             Catalog name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                       Help for analytics:create
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --query string               Add query to request
      --scope string               scope
  -s, --server string              management server endpoint (required)
      --space string               Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services

Analytics Services operations

Synopsis

Analytics Services operations

```
apic analytics-services [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
  -c, --catalog string             Catalog name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                       Help for analytics-services
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
      --space string               Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:clear

Clear the Analytics Service objects

Synopsis

Clear the Analytics Service objects

```
apic analytics-services:clear [flags]
```

Possible Usages

```
    apic analytics-services:clear [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --confirm string             Confirmation for critical updates (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:create

Create a Analytics Service object

Synopsis

Create a Analytics Service object

```
      Fields not allowed:
      - gateway_service_urls
      - ingestion_endpoint
      - client_endpoint

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic analytics-services:create [flags]
```

Possible Usages

```
      apic analytics-services:create [flags] ANALYTICS_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:delete

Delete the Analytics Service object by name or id

Synopsis

Delete the Analytics Service object by name or id

```
apic analytics-services:delete [flags]
```

Possible Usages

```
      apic analytics-services:delete [flags] ANALYTICS_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:get

Get the Analytics Service object by name or id

Synopsis

Get the Analytics Service object by name or id

```
apic analytics-services:get [flags]
```

Possible Usages

```
      apic analytics-services:get [flags] --scope space ANALYTICS_SERVICE
      apic analytics-services:get [flags] --scope org ANALYTICS_SERVICE
      apic analytics-services:get [flags] --scope catalog ANALYTICS_SERVICE
      apic analytics-services:get [flags] ANALYTICS_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:list

List the Analytics Service objects

Synopsis

List the Analytics Service objects

```
apic analytics-services:list [flags]
```

Possible Usages

```
      apic analytics-services:list [flags] --scope space
      apic analytics-services:list [flags] --scope org
      apic analytics-services:list [flags] --scope catalog
      apic analytics-services:list [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic analytics-services:update

Update the Analytics Service object by name or id

Synopsis

Update the Analytics Service object by name or id

```
      Fields not allowed to be null:
      - ingestion_endpoint_tls_client_profile_url
      - client_endpoint_tls_client_profile_url

      Fields not allowed:
      - endpoint
      - ingestion_endpoint
      - client_endpoint

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic analytics-services:update [flags]
```

Possible Usages

```
      apic analytics-services:update [flags] ANALYTICS_SERVICE ANALYTICS_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic api-keys

Api Keys operations

Synopsis

Api Keys operations

```
apic api-keys [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for api-keys
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic api-keys:create

Create a API Key object

Synopsis

Create a API Key object

```
      Required fields:
      - client_type
      - description

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic api-keys:create [flags]
```

Possible Usages

```
      apic api-keys:create [flags] API_KEY_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api-keys:delete

Delete the API Key object by name or id

Synopsis

Delete the API Key object by name or id

```
apic api-keys:delete [flags]
```

Possible Usages

```
      apic api-keys:delete [flags] API_KEY
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api-keys:get

Get the API Key object by name or id

Synopsis

Get the API Key object by name or id

```
apic api-keys:get [flags]
```

Possible Usages

```
apic api-keys:get [flags] API_KEY
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api-keys:list

List the API Key objects

Synopsis

List the API Key objects

```
apic api-keys:list [flags]
```

Possible Usages

```
apic api-keys:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis

Apis operations

Synopsis

Apis operations

```
apic apis [flags]
```

Options

```
      --api_type string    The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --base_path string   Base path of the API
  -c, --catalog string     Catalog name or id (required)
      --enforced           Whether the API is enforced or not
      --expand string      List of transient field to expand
      --fields string      List of field names to return
      --format string      Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help               Help for apis
      --limit int32        Maximum number of items to return
      --metadata string    List of metadata fields in the api to filter on
```

```
      --offset int32        Offset item number from list to begin return
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --space string        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis:clone

Apis clone operations

Synopsis

Apis clone operations

```
apic apis:clone [flags]
```

Possible Usages

```
apic apis:clone [flags] --scope space
apic apis:clone [flags] --scope catalog
```

Options

```
      --api_type string    The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --base_path string   Base path of the API
  -c, --catalog string     Catalog name or id (required)
      --enforced           Whether the API is enforced or not
      --expand string      List of transient field to expand
      --fields string      List of field names to return
      --format string      Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help               Help for apis:clone
      --limit int32        Maximum number of items to return
      --metadata string    List of metadata fields in the api to filter on
      --offset int32       Offset item number from list to begin return
  -o, --org string         Organization name or id (required)
      --output string      Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string       scope
  -s, --server string      management server endpoint (required)
      --space string       Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis:document

Apis document operations

Synopsis

Apis document operations

```
apic apis:document [flags]
```

Possible Usages

```
apic apis:document [flags] --id --scope space ID
apic apis:document [flags] --id --scope catalog ID
apic apis:document [flags] --scope space NAME:VERSION
apic apis:document [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for apis:document
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis:get

Apis get operations

Synopsis

Apis get operations

```
apic apis:get [flags]
```

Possible Usages

```
    apic apis:get [flags] --id --scope space ID
    apic apis:get [flags] --id --scope catalog ID
    apic apis:get [flags] --scope space NAME:VERSION
    apic apis:get [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return (default "add(wsdl,api)")
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for apis:get
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis:list

Apis list operations

Synopsis

Apis list operations

```
apic apis:list [flags]
```

Possible Usages

```
    apic apis:list [flags] --scope space API_NAME
    apic apis:list [flags] --scope catalog API_NAME
```

Options

```
      --api_type string    The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --base_path string   Base path of the API
  -c, --catalog string     Catalog name or id (required)
```

```
      --enforced            Whether the API is enforced or not
      --expand string       List of transient field to expand
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for apis:list
      --limit int32         Maximum number of items to return
      --metadata string     List of metadata fields in the api to filter on
      --offset int32        Offset item number from list to begin return
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --space string        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic apis:list-all

Apis list-all operations

Synopsis

Apis list-all operations

```
apic apis:list-all [flags]
```

Possible Usages

```
      apic apis:list-all [flags] --scope space
      apic apis:list-all [flags] --scope catalog
```

Options

```
      --api_type string     The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --base_path string    Base path of the API
  -c, --catalog string      Catalog name or id (required)
      --enforced            Whether the API is enforced or not
      --expand string       List of transient field to expand
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for apis:list-all
      --limit int32         Maximum number of items to return
      --metadata string     List of metadata fields in the api to filter on
      --offset int32        Offset item number from list to begin return
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --space string        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic apis:update

Apis update operations

Synopsis

Apis update operations

```
apic apis:update [flags]
```

Possible Usages

```
      apic apis:update [flags] --id --scope space ID API_FILE
      apic apis:update [flags] --id --scope catalog ID API_FILE
      apic apis:update [flags] --scope space NAME:VERSION API_FILE
      apic apis:update [flags] --scope catalog NAME:VERSION API_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for apis:update
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apis:wsdl

Apis wsdl operations

Synopsis

Apis wsdl operations

```
apic apis:wsdl [flags]
```

Possible Usages

```
      apic apis:wsdl [flags] --id --scope space ID
      apic apis:wsdl [flags] --id --scope catalog ID
      apic apis:wsdl [flags] --scope space NAME:VERSION
      apic apis:wsdl [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for apis:wsdl
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic apps

Apps operations

Synopsis

Apps operations

```
apic apps [flags]
```

Options

```
  -c, --catalog string        Catalog name or id (required)
      --consumer-org string   Consumer Organization name or id (required)
```

```
      --expand string          List of transient field to expand
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for apps
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:clear

Clear the Application objects

Synopsis

Clear the Application objects

```
apic apps:clear [flags]
```

Possible Usages

```
      apic apps:clear [flags] --space-initiated
      apic apps:clear [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --confirm string         Confirmation for critical updates (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:create

Create a Application object

Synopsis

Create a Application object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
```

```
apic apps:create [flags]
```

Possible Usages

```
      apic apps:create [flags] --space-initiated APP_FILE
      apic apps:create [flags] APP_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:delete

Delete the Application object by name or id

Synopsis

Delete the Application object by name or id

```
apic apps:delete [flags]
```

Possible Usages

```
      apic apps:delete [flags] --space-initiated APP
      apic apps:delete [flags] APP
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:get

Get the Application object by name or id

Synopsis

Get the Application object by name or id

```
apic apps:get [flags]
```

Possible Usages

```
      apic apps:get [flags] --space-initiated APP
      apic apps:get [flags] APP
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:list

List the Application objects

Synopsis

List the Application objects

```
apic apps:list [flags]
```

Possible Usages

```
      apic apps:list [flags] --space-initiated
      apic apps:list [flags] --scope space
      apic apps:list [flags] --scope catalog
      apic apps:list [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apps:update

Update the Application object by name or id

Synopsis

Update the Application object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
```

```
apic apps:update [flags]
```

Possible Usages

```
      apic apps:update [flags] --space-initiated APP APP_FILE
      apic apps:update [flags] APP APP_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic associates

Associates operations

Synopsis

Associates operations

```
apic associates [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for associates
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic associates:get

Associates get operations

Synopsis

Associates get operations

```
apic associates:get [flags]
```

Possible Usages

```
      apic associates:get [flags] --scope consumer-org --space-initiated ASSOCIATE
      apic associates:get [flags] --scope org ASSOCIATE
      apic associates:get [flags] --scope consumer-org ASSOCIATE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
```

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for associates:get
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic associates:list

Associates list operations

Synopsis

Associates list operations

```
apic associates:list [flags]
```

Possible Usages

```
      apic associates:list [flags] --scope consumer-org --space-initiated
      apic associates:list [flags] --scope org
      apic associates:list [flags] --scope consumer-org
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for associates:list
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic availability-zones

Availability Zones operations

Synopsis

Availability Zones operations

```
apic availability-zones [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for availability-zones
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
```

```
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:clear

Clear the Availability Zone objects

Synopsis

Clear the Availability Zone objects

```
apic availability-zones:clear [flags]
```

Possible Usages

```
      apic availability-zones:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:create

Create a Availability Zone object

Synopsis

Create a Availability Zone object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic availability-zones:create [flags]
```

Possible Usages

```
      apic availability-zones:create [flags] AVAILABILITY_ZONE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:delete

Delete the Availability Zone object by name or id

Synopsis

Delete the Availability Zone object by name or id

```
apic availability-zones:delete [flags]
```

Possible Usages

```
      apic availability-zones:delete [flags] AVAILABILITY_ZONE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:get

Get the Availability Zone object by name or id

Synopsis

Get the Availability Zone object by name or id

```
apic availability-zones:get [flags]
```

Possible Usages

```
      apic availability-zones:get [flags] AVAILABILITY_ZONE
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:list

List the Availability Zone objects

Synopsis

List the Availability Zone objects

```
apic availability-zones:list [flags]
```

Possible Usages

```
apic availability-zones:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic availability-zones:update

Update the Availability Zone object by name or id

Synopsis

Update the Availability Zone object by name or id

```
Fields allowed but ignored:
- id
- type
- api_version
- scope
- created_at
- updated_at
- url
- org_url
```

```
apic availability-zones:update [flags]
```

Possible Usages

```
apic availability-zones:update [flags] AVAILABILITY_ZONE AVAILABILITY_ZONE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings

Billings operations

Synopsis

Billings operations

```
apic billings [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for billings
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings:clear

Clear the Billing objects

Synopsis

Clear the Billing objects

```
apic billings:clear [flags]
```

Possible Usages

```
      apic billings:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings:delete

Delete the Billing object by name or id

Synopsis

Delete the Billing object by name or id

```
apic billings:delete [flags]
```

Possible Usages

```
      apic billings:delete [flags] BILLING
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings:get

Get the Billing object by name or id

Synopsis

Get the Billing object by name or id

```
apic billings:get [flags]
```

Possible Usages

```
      apic billings:get [flags] BILLING
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings:list

List the Billing objects

Synopsis

List the Billing objects

```
apic billings:list [flags]
```

Possible Usages

```
      apic billings:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic billings:update

Update the Billing object by name or id

Synopsis

Update the Billing object by name or id

```
      Fields not allowed to be null:
      - integration_url

      Fields not allowed:
      - job_queue_status

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic billings:update [flags]
```

Possible Usages

```
      apic billings:update [flags] BILLING BILLING_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalog-settings

Catalog Settings operations

Synopsis

Catalog Settings operations

```
apic catalog-settings [flags]
```

Options

```
  -h, --help   Help for catalog-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalog-settings:get

Get the Catalog Setting object

Synopsis

Get the Catalog Setting object

```
apic catalog-settings:get [flags]
```

Possible Usages

```
      apic catalog-settings:get [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalog-settings:update

Update the Catalog Setting object

Synopsis

Update the Catalog Setting object

```
      Fields allowed but ignored:
      - name
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
```

```
apic catalog-settings:update [flags]
```

Possible Usages

```
      apic catalog-settings:update [flags] CATALOG_SETTING_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --force            Force the operation
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs

Catalogs operations

Synopsis

Catalogs operations

```
apic catalogs [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for catalogs
      --limit int32     Maximum number of items to return
      --my              my
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:clear

Clear the Catalog objects

Synopsis

Clear the Catalog objects

```
apic catalogs:clear [flags]
```

Possible Usages

```
      apic catalogs:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:create

Create a Catalog object

Synopsis

Create a Catalog object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic catalogs:create [flags]
```

Possible Usages

```
      apic catalogs:create [flags] CATALOG_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic catalogs:delete

Delete the Catalog object by name or id

Synopsis

Delete the Catalog object by name or id

```
apic catalogs:delete [flags]
```

Possible Usages

```
      apic catalogs:delete [flags] CATALOG
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:email-to-owners

Send email to owners of consumer organizations

Synopsis

Send email to owners of consumer organizations, given consumer org and consumer group urls. For consumer group, email owners of all consumer orgs in the group.

```
apic catalogs:email-to-owners [flags]
```

Possible Usages

```
      apic catalogs:email-to-owners [flags] CATALOG EMAIL_TO_OWNERS_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:get

Get the Catalog object by name or id

Synopsis

Get the Catalog object by name or id

```
apic catalogs:get [flags]
```

Possible Usages

```
      apic catalogs:get [flags] CATALOG
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:list

List the Catalog objects

Synopsis

List the Catalog objects

```
apic catalogs:list [flags]
```

Possible Usages

```
      apic catalogs:list [flags] --my
      apic catalogs:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:transfer-owner

Transfer owner to an associate

Synopsis

Transfer owner to an associate

```
apic catalogs:transfer-owner [flags]
```

Possible Usages

```
      apic catalogs:transfer-owner [flags] CATALOG TRANSFER_OWNER_FILE
```

Options

```
      --cascade            Cascade the behavior
      --delete_old_owner   Delete old owner
      --format string      Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string         Organization name or id (required)
```

```
      --output string      Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string      management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic catalogs:update

Update the Catalog object by name or id

Synopsis

Update the Catalog object by name or id

```
      Fields not allowed:
      - name

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic catalogs:update [flags]
```

Possible Usages

```
      apic catalogs:update [flags] CATALOG CATALOG_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic client-creds

Manage the client configuration parameters. Client ID and Client Secret.

Synopsis

Manage the client configuration parameters. Client ID and Client Secret.

```
apic client-creds [flags]
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for client-creds
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic client-creds:clear

clear the set client credentials.

Synopsis

clear the set client credentials.

```
apic client-creds:clear [flags]
```

Examples

```
$ apic client-creds:clear
Deleted client credentials.
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for client-creds:clear
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic client-creds:list

List the set clientID/secret parameters.

Synopsis

List the set clientID/secret parameters.

```
apic client-creds:list [flags]
```

Examples

```
$ apic client-creds:list
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for client-creds:list
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic client-creds:set

Set the client configuration parameters. ClientID and Client Secret.

Synopsis

Set the client configuration parameters. ClientID and Client Secret.

```
apic client-creds:set /path/to/creds/json ... [flags]
```

Examples

```
$ apic client-creds:set /Users/local_user/credential.json
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for client-creds:set
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-emails

Cloud Emails operations

Synopsis

Cloud Emails operations

```
apic cloud-emails [flags]
```

Options

```
  -h, --help   Help for cloud-emails
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-emails:send-to-owners

Email provider org owners

Synopsis

Send an email to owners of provider orgs

```
apic cloud-emails:send-to-owners [flags]
```

Possible Usages

```
    apic cloud-emails:send-to-owners [flags] EMAIL_PROVIDER_ORG_OWNERS_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings

Cloud Settings operations

Synopsis

Cloud Settings operations

```
apic cloud-settings [flags]
```

Options

```
  -h, --help   Help for cloud-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:about

Return information about the cloud

Synopsis

Return public information about the cloud. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:about [flags]
```

Possible Usages

```
      apic cloud-settings:about [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:audit-endpoint-test-connection

Test an audit endpoint connection

Synopsis

Test an audit endpoint connection

```
apic cloud-settings:audit-endpoint-test-connection [flags]
```

Possible Usages

```
      apic cloud-settings:audit-endpoint-test-connection [flags] ENDPOINT
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:designer-credentials-list

List credential for designer

Synopsis

List credential for designer

```
apic cloud-settings:designer-credentials-list [flags]
```

Possible Usages

```
      apic cloud-settings:designer-credentials-list [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:get

Get the Cloud Setting object

Synopsis

Get the Cloud Setting object

```
apic cloud-settings:get [flags]
```

Possible Usages

```
      apic cloud-settings:get [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:info

Return public information about the cloud deployment

Synopsis

Return public information about the cloud deployment

```
apic cloud-settings:info [flags]
```

Possible Usages

```
      apic cloud-settings:info [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:mail-server-configured

Return true or false based on if mail server is configured or not

Synopsis

Return true or false based on if mail server is configured or not. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:mail-server-configured [flags]
```

Possible Usages

```
    apic cloud-settings:mail-server-configured [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:oauth2-certs

Support JWKS_URI endpoint, conform to OIDC specification

Synopsis

Return JWKS that is used to issue token. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:oauth2-certs [flags]
```

Possible Usages

```
    apic cloud-settings:oauth2-certs [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:toolkit-credentials-list

List credential for toolkit, consumer toolkit, governance, and discovery toolkit

Synopsis

List credential for toolkit, consumer toolkit, governance, and discovery toolkit

```
apic cloud-settings:toolkit-credentials-list [flags]
```

Possible Usages

```
apic cloud-settings:toolkit-credentials-list [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:topology

Return the topology of the cloud for governance

Synopsis

Return the topology of the cloud for governance. You MUST supply the --format option when using this command from the API Connect toolkit.

```
apic cloud-settings:topology [flags]
```

Possible Usages

```
apic cloud-settings:topology [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic cloud-settings:update

Update the Cloud Setting object

Synopsis

Update the Cloud Setting object

```
Fields not allowed to be null:
- email_sender
- tls_client_profile_default_url

Fields not allowed:
- service_type
- cloud_id
- ibm_cloud

Fields allowed but ignored:
- name
- type
- api_version
```

```
      - scope
      - created_at
      - updated_at
      - url
```

```
apic cloud-settings:update [flags]
```

Possible Usages

```
      apic cloud-settings:update [flags] CLOUD_SETTING_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config

Manage configuration variables

Synopsis

Manage configuration variables

```
apic config [flags]
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:clear

Delete all configuration variables

Synopsis

Delete all configuration variables

```
apic config:clear [flags]
```

Examples

```
$ apic config:clear
Deleted config catalog
Deleted config org
Deleted config space

$ apic config:clear --global
Deleted config catalog
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:clear
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic config:delete

Delete a configuration variable

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:delete NAME... [flags]
```

Examples

```
$ apic config:delete catalog
Deleted config catalog

$ apic config:delete org
Deleted config org

$ apic config:delete template-path
Deleted config template-path
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:delete
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic config:get

Get a configuration variable

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:get NAME [flags]
```

Examples

```
$ apic config:get catalog
catalog: https://mgmnthost.com/api/catalogs/climbon/sb

$ apic config:get org
org: https://mgmnthost.com/api/orgs/climbon

$ apic config:get template-path
template-path: /etc/templates
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:get
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:list

List the application and global configuration variables

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:list [flags]
```

Examples

```
$ apic config
catalog: https://mgmnthost.com/api/catalogs/climbon/sb
org: https://mgmnthost.com/api/orgs/climbon

$ apic config --global
catalog: https://mgmnthost.com/api/catlogs/climbon/sb
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:list
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:set

Set or update configuration variables

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/api/catalogs/ORGANIZATION_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/api/orgs/ORGANIZATION_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/api/spaces/ORGANIZATION_NAME/CATALOG_NAME/SPACE_NAME

consumer The consumer configuration variable should be set to the URI of an API Connect consumer. The value of the consumer-org provides the default identity of a consumer. The default values defined by the consumer URI can be overridden using the --server, --organization, --catalog, and --consumer command line options. The consumer URI has the form: https://MANAGEMENT_SERVER/api/consumer-orgs/ORGANIZATION_NAME/CATALOG_NAME/CONSUMER_ORG_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:set NAME=VALUE ... [flags]
```

Examples

```
$ apic config:set catalog=https://mgmnthost.com/api/catalogs/climbon/sb
catalog: https://mgmnthost.com/api/catalogs/climbon/sb

$ apic config:set org=https://mgmnthost2.com/api/orgs/hikeon
org: https://mgmnthost2.com/api/orgs/hikeon

$ apic config:set --global template-path="/etc/templates"
template-path: /etc/templates
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:set
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries

Configured Api User Registries operations

Synopsis

Configured Api User Registries operations

```
apic configured-api-user-registries [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries:clear

Configured Api User Registries clear operations

Synopsis

Configured Api User Registries clear operations

```
apic configured-api-user-registries:clear [flags]
```

Possible Usages

```
    apic configured-api-user-registries:clear [flags] --scope space
    apic configured-api-user-registries:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries:create

Configured Api User Registries create operations

Synopsis

Configured Api User Registries create operations

```
apic configured-api-user-registries:create [flags]
```

Possible Usages

```
      apic configured-api-user-registries:create [flags] --scope space CONFIGURED_API_USER_REGISTRY_FILE
      apic configured-api-user-registries:create [flags] --scope catalog CONFIGURED_API_USER_REGISTRY_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries:create
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries:delete

Configured Api User Registries delete operations

Synopsis

Configured Api User Registries delete operations

```
apic configured-api-user-registries:delete [flags]
```

Possible Usages

```
      apic configured-api-user-registries:delete [flags] --scope space CONFIGURED_API_USER_REGISTRY
      apic configured-api-user-registries:delete [flags] --scope catalog CONFIGURED_API_USER_REGISTRY
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries:delete
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries:get

Configured Api User Registries get operations

Synopsis

Configured Api User Registries get operations

```
apic configured-api-user-registries:get [flags]
```

Possible Usages

```
apic configured-api-user-registries:get [flags] --scope space CONFIGURED_API_USER_REGISTRY
apic configured-api-user-registries:get [flags] --scope catalog CONFIGURED_API_USER_REGISTRY
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-api-user-registries:list

Configured Api User Registries list operations

Synopsis

Configured Api User Registries list operations

```
apic configured-api-user-registries:list [flags]
```

Possible Usages

```
apic configured-api-user-registries:list [flags] --scope space
apic configured-api-user-registries:list [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-api-user-registries:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings

Configured Billings operations

Synopsis

Configured Billings operations

```
apic configured-billings [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-billings
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings:clear

Clear the Configured Billing objects

Synopsis

Clear the Configured Billing objects

```
apic configured-billings:clear [flags]
```

Possible Usages

```
    apic configured-billings:clear [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings:create

Create a Configured Billing object

Synopsis

Create a Configured Billing object

```
    Required fields:
    - billing_url

    Fields allowed but ignored:
    - id
    - type
    - api_version
    - scope
    - created_at
```

```
      - updated_at
      - url
      - org_url
      - catalog_url
```

```
apic configured-billings:create [flags]
```

Possible Usages

```
      apic configured-billings:create [flags] CONFIGURED_BILLING_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings:delete

Delete the Configured Billing object by name or id

Synopsis

Delete the Configured Billing object by name or id

```
apic configured-billings:delete [flags]
```

Possible Usages

```
      apic configured-billings:delete [flags] CONFIGURED_BILLING
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings:get

Get the Configured Billing object by name or id

Synopsis

Get the Configured Billing object by name or id

```
apic configured-billings:get [flags]
```

Possible Usages

```
      apic configured-billings:get [flags] CONFIGURED_BILLING
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-billings:list

List the Configured Billing objects

Synopsis

List the Configured Billing objects

```
apic configured-billings:list [flags]
```

Possible Usages

```
      apic configured-billings:list [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries

Configured Catalog User Registries operations

Synopsis

Configured Catalog User Registries operations

```
apic configured-catalog-user-registries [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-catalog-user-registries
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries:create

Create a Configured Catalog User Registry object

Synopsis

Create a Configured Catalog User Registry object

```
      Required fields:
      - user_registry_url

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - user_registry_url
```

```
apic configured-catalog-user-registries:create [flags]
```

Possible Usages

```
      apic configured-catalog-user-registries:create [flags] CONFIGURED_CATALOG_USER_REGISTRY_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries:delete

Delete the Configured Catalog User Registry object by name or id

Synopsis

Delete the Configured Catalog User Registry object by name or id

```
apic configured-catalog-user-registries:delete [flags]
```

Possible Usages

```
      apic configured-catalog-user-registries:delete [flags] CONFIGURED_CATALOG_USER_REGISTRY
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries:get

Get the Configured Catalog User Registry object by name or id

Synopsis

Get the Configured Catalog User Registry object by name or id

```
apic configured-catalog-user-registries:get [flags]
```

Possible Usages

```
      apic configured-catalog-user-registries:get [flags] CONFIGURED_CATALOG_USER_REGISTRY
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries:list

List the Configured Catalog User Registry objects

Synopsis

List the Configured Catalog User Registry objects

```
apic configured-catalog-user-registries:list [flags]
```

Possible Usages

```
      apic configured-catalog-user-registries:list [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic configured-catalog-user-registries:search

Search for users in the catalog user registry

Synopsis

Search for users in the catalog user registry

```
apic configured-catalog-user-registries:search [flags]
```

Possible Usages

```
    apic configured-catalog-user-registries:search [flags] CONFIGURED_CATALOG_USER_REGISTRY USER_SEARCH_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services

Configured Gateway Services operations

Synopsis

Configured Gateway Services operations

```
apic configured-gateway-services [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services:clear

Configured Gateway Services clear operations

Synopsis

Configured Gateway Services clear operations

```
apic configured-gateway-services:clear [flags]
```

Possible Usages

```
    apic configured-gateway-services:clear [flags] --scope space
    apic configured-gateway-services:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services:create

Configured Gateway Services create operations

Synopsis

Configured Gateway Services create operations

```
apic configured-gateway-services:create [flags]
```

Possible Usages

```
    apic configured-gateway-services:create [flags] --scope space CONFIGURED_GATEWAY_SERVICE_FILE
    apic configured-gateway-services:create [flags] --scope catalog CONFIGURED_GATEWAY_SERVICE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services:create
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services:delete

Configured Gateway Services delete operations

Synopsis

Configured Gateway Services delete operations

```
apic configured-gateway-services:delete [flags]
```

Possible Usages

```
    apic configured-gateway-services:delete [flags] --scope space CONFIGURED_GATEWAY_SERVICE
    apic configured-gateway-services:delete [flags] --scope catalog CONFIGURED_GATEWAY_SERVICE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services:delete
  -o, --org string       Organization name or id (required)
```

```
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services:get

Configured Gateway Services get operations

Synopsis

Configured Gateway Services get operations

```
apic configured-gateway-services:get [flags]
```

Possible Usages

```
      apic configured-gateway-services:get [flags] --scope space CONFIGURED_GATEWAY_SERVICE
      apic configured-gateway-services:get [flags] --scope catalog CONFIGURED_GATEWAY_SERVICE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-gateway-services:list

Configured Gateway Services list operations

Synopsis

Configured Gateway Services list operations

```
apic configured-gateway-services:list [flags]
```

Possible Usages

```
      apic configured-gateway-services:list [flags] --scope space
      apic configured-gateway-services:list [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-gateway-services:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers

Configured Oauth Providers operations

Synopsis

Configured Oauth Providers operations

```
apic configured-oauth-providers [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers:clear

Configured Oauth Providers clear operations

Synopsis

Configured Oauth Providers clear operations

```
apic configured-oauth-providers:clear [flags]
```

Possible Usages

```
    apic configured-oauth-providers:clear [flags] --scope space
    apic configured-oauth-providers:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers:create

Configured Oauth Providers create operations

Synopsis

Configured Oauth Providers create operations

```
apic configured-oauth-providers:create [flags]
```

Possible Usages

```
      apic configured-oauth-providers:create [flags] --scope space CONFIGURED_OAUTH_PROVIDER_FILE
      apic configured-oauth-providers:create [flags] --scope catalog CONFIGURED_OAUTH_PROVIDER_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers:create
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers:delete

Configured Oauth Providers delete operations

Synopsis

Configured Oauth Providers delete operations

```
apic configured-oauth-providers:delete [flags]
```

Possible Usages

```
      apic configured-oauth-providers:delete [flags] --scope space CONFIGURED_OAUTH_PROVIDER
      apic configured-oauth-providers:delete [flags] --scope catalog CONFIGURED_OAUTH_PROVIDER
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers:delete
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers:get

Configured Oauth Providers get operations

Synopsis

Configured Oauth Providers get operations

```
apic configured-oauth-providers:get [flags]
```

Possible Usages

```
apic configured-oauth-providers:get [flags] --scope space CONFIGURED_OAUTH_PROVIDER
apic configured-oauth-providers:get [flags] --scope catalog CONFIGURED_OAUTH_PROVIDER
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-oauth-providers:list

Configured Oauth Providers list operations

Synopsis

Configured Oauth Providers list operations

```
apic configured-oauth-providers:list [flags]
```

Possible Usages

```
apic configured-oauth-providers:list [flags] --scope space
apic configured-oauth-providers:list [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-oauth-providers:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles

Configured Tls Client Profiles operations

Synopsis

Configured Tls Client Profiles operations

```
apic configured-tls-client-profiles [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:clear

Configured Tls Client Profiles clear operations

Synopsis

Configured Tls Client Profiles clear operations

```
apic configured-tls-client-profiles:clear [flags]
```

Possible Usages

```
    apic configured-tls-client-profiles:clear [flags] --scope space CONFIGURED_TLS_CLIENT_PROFILE_NAME
    apic configured-tls-client-profiles:clear [flags] --scope catalog CONFIGURED_TLS_CLIENT_PROFILE_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:clear-all

Configured Tls Client Profiles clear-all operations

Synopsis

Configured Tls Client Profiles clear-all operations

```
apic configured-tls-client-profiles:clear-all [flags]
```

Possible Usages

```
    apic configured-tls-client-profiles:clear-all [flags] --scope space
    apic configured-tls-client-profiles:clear-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:clear-all
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:create

Configured Tls Client Profiles create operations

Synopsis

Configured Tls Client Profiles create operations

```
apic configured-tls-client-profiles:create [flags]
```

Possible Usages

```
    apic configured-tls-client-profiles:create [flags] --scope space CONFIGURED_TLS_CLIENT_PROFILE_FILE
    apic configured-tls-client-profiles:create [flags] --scope catalog CONFIGURED_TLS_CLIENT_PROFILE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:create
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:delete

Configured Tls Client Profiles delete operations

Synopsis

Configured Tls Client Profiles delete operations

```
apic configured-tls-client-profiles:delete [flags]
```

Possible Usages

```
    apic configured-tls-client-profiles:delete [flags] --id --scope space ID
    apic configured-tls-client-profiles:delete [flags] --id --scope catalog ID
    apic configured-tls-client-profiles:delete [flags] --scope space NAME:VERSION
    apic configured-tls-client-profiles:delete [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:delete
      --id               id
  -o, --org string       Organization name or id (required)
```

```
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:get

Configured Tls Client Profiles get operations

Synopsis

Configured Tls Client Profiles get operations

```
apic configured-tls-client-profiles:get [flags]
```

Possible Usages

```
      apic configured-tls-client-profiles:get [flags] --id --scope space ID
      apic configured-tls-client-profiles:get [flags] --id --scope catalog ID
      apic configured-tls-client-profiles:get [flags] --scope space NAME:VERSION
      apic configured-tls-client-profiles:get [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:get
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:list

Configured Tls Client Profiles list operations

Synopsis

Configured Tls Client Profiles list operations

```
apic configured-tls-client-profiles:list [flags]
```

Possible Usages

```
      apic configured-tls-client-profiles:list [flags] --scope space CONFIGURED_TLS_CLIENT_PROFILE_NAME
      apic configured-tls-client-profiles:list [flags] --scope catalog CONFIGURED_TLS_CLIENT_PROFILE_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
```

```
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic configured-tls-client-profiles:list-all

Configured Tls Client Profiles list-all operations

Synopsis

Configured Tls Client Profiles list-all operations

```
apic configured-tls-client-profiles:list-all [flags]
```

Possible Usages

```
      apic configured-tls-client-profiles:list-all [flags] --scope space
      apic configured-tls-client-profiles:list-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for configured-tls-client-profiles:list-all
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-org-settings

Consumer Org Settings operations

Synopsis

Consumer Org Settings operations

```
apic consumer-org-settings [flags]
```

Options

```
  -h, --help   Help for consumer-org-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-org-settings:delete

Delete the Consumer Organization Setting object

Synopsis

Delete the Consumer Organization Setting object

```
apic consumer-org-settings:delete [flags]
```

Possible Usages

```
      apic consumer-org-settings:delete [flags] --space-initiated
      apic consumer-org-settings:delete [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic consumer-org-settings:get

Get the Consumer Organization Setting object

Synopsis

Get the Consumer Organization Setting object

```
apic consumer-org-settings:get [flags]
```

Possible Usages

```
      apic consumer-org-settings:get [flags] --space-initiated
      apic consumer-org-settings:get [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic consumer-org-settings:update

Update the Consumer Organization Setting object

Synopsis

Update the Consumer Organization Setting object

```
      Fields allowed but ignored:
      - name
```

```
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic consumer-org-settings:update [flags]
```

Possible Usages

```
      apic consumer-org-settings:update [flags] --space-initiated CONSUMER_ORG_SETTING_FILE
      apic consumer-org-settings:update [flags] CONSUMER_ORG_SETTING_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic consumer-orgs

Consumer Orgs operations

Synopsis

Consumer Orgs operations

```
apic consumer-orgs [flags]
```

Options

```
  -c, --catalog string     Catalog name or id (required)
      --expand string      List of transient field to expand
      --fields string      List of field names to return
      --format string      Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help               Help for consumer-orgs
      --limit int32        Maximum number of items to return
      --offset int32       Offset item number from list to begin return
  -o, --org string         Organization name or id (required)
      --output string      Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string      management server endpoint (required)
      --space string       Space name or id (required)
      --space-initiated    space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic consumer-orgs:clear

Clear the Consumer Organization objects

Synopsis

Clear the Consumer Organization objects

```
apic consumer-orgs:clear [flags]
```

Possible Usages

```
      apic consumer-orgs:clear [flags] --space-initiated
      apic consumer-orgs:clear [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:create

Create a Consumer Organization object

Synopsis

Create a Consumer Organization object

```
      Required fields:
      - owner_url

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
```

```
apic consumer-orgs:create [flags]
```

Possible Usages

```
      apic consumer-orgs:create [flags] --space-initiated CONSUMER_ORG_FILE
      apic consumer-orgs:create [flags] CONSUMER_ORG_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:delete

Delete the Consumer Organization object by name or id

Synopsis

Delete the Consumer Organization object by name or id

```
apic consumer-orgs:delete [flags]
```

Possible Usages

```
apic consumer-orgs:delete [flags] --space-initiated CONSUMER_ORG
apic consumer-orgs:delete [flags] CONSUMER_ORG
```

Options

```
-c, --catalog string   Catalog name or id (required)
    --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
-o, --org string       Organization name or id (required)
    --output string    Write file(s) to directory, instead of STDOUT (default "-")
-s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
    --accept-license        Accept the license for API Connect
    --debug                 Enable debug output
    --debug-output string   Write debug output to file
    --live-help             Enable or disable tracking of limited usage information
-m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:get

Get the Consumer Organization object by name or id

Synopsis

Get the Consumer Organization object by name or id

```
apic consumer-orgs:get [flags]
```

Possible Usages

```
apic consumer-orgs:get [flags] --space-initiated CONSUMER_ORG
apic consumer-orgs:get [flags] CONSUMER_ORG
```

Options

```
-c, --catalog string   Catalog name or id (required)
    --fields string    List of field names to return
    --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
-o, --org string       Organization name or id (required)
    --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
-s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
    --accept-license        Accept the license for API Connect
    --debug                 Enable debug output
    --debug-output string   Write debug output to file
    --live-help             Enable or disable tracking of limited usage information
-m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:list

List the Consumer Organization objects

Synopsis

List the Consumer Organization objects

```
apic consumer-orgs:list [flags]
```

Possible Usages

```
apic consumer-orgs:list [flags] --space-initiated
apic consumer-orgs:list [flags]
```

Options

```
-c, --catalog string   Catalog name or id (required)
    --expand string    List of transient field to expand
    --fields string    List of field names to return
    --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
```

```
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:transfer-owner

Transfer owner to an associate

Synopsis

Transfer owner to an associate

```
apic consumer-orgs:transfer-owner [flags]
```

Possible Usages

```
      apic consumer-orgs:transfer-owner [flags] --space-initiated CONSUMER_ORG TRANSFER_OWNER_FILE
      apic consumer-orgs:transfer-owner [flags] CONSUMER_ORG TRANSFER_OWNER_FILE
```

Options

```
      --cascade             Cascade the behavior
  -c, --catalog string      Catalog name or id (required)
      --delete_old_owner    Delete old owner
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string       management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-orgs:update

Update the Consumer Organization object by name or id

Synopsis

Update the Consumer Organization object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url

apic consumer-orgs:update [flags]
```

Possible Usages

```
      apic consumer-orgs:update [flags] --space-initiated CONSUMER_ORG CONSUMER_ORG_FILE
      apic consumer-orgs:update [flags] CONSUMER_ORG CONSUMER_ORG_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic create

Create an API or product definition

Synopsis

Create an API or product definition

```
apic create [flags]
```

Options

```
  -h, --help          Help for create
  -i, --interactive   use interactive mode
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic create:api

Create an OpenAPI (Swagger) definition

Synopsis

Create an OpenAPI (Swagger) definition

```
apic create:api [flags]
```

Examples

```
Create an API
  $ apic create:api --title Routes
  Created routes.yaml API definition [routes:1.0.0]

Create an API and generate a product referencing the API
  $ apic create:api --title Routes --product "Climb On"
  Created routes.yaml API definition [routes:1.0.0]
  Created climb-on.yaml product definition [climb-on:1.0.0]

Create an API interactively
  $ apic create:api
  Title: Routes
  Name (routes): routes
  File (routes.yaml): routes.yaml
  Template: ()
  Basepath (/routes): /routes
  Hostname ($(catalog.host)): $(catalog.host)
  Schemes:
  Target url: ()
  Create product [true]: true
  Product title (Routes Product): Climb On
  Product name (climb-on): climb-on
  Product file (climb-on.yaml): climb-on.yaml
  Created routes.yaml API definition [routes:1.0.0]
  Created climb-on.yaml product definition [climb-on:1.0.0]

Create an API from a WSDL document
  $ apic create:api --wsdl globalweather.wsdl
  Created globalweather.yaml API definition [globalweather.yaml:1.0.0]
```

```
Create an API using APIC's default OAuth 2 provider template
  $ apic create:api --title "OAuth2 Provider" --template oauth2
  Created oauth2-provider.yaml API definition [oauth2-provider:1.0.0]

Create an API using one of your templates
  $ apic config:set --global template-path="/etc/templates"
  $ ls /etc/templates
  proxy.hbs staging.hbs
  $ apic create:api --title "Proxy Provider" --template proxy
  Created proxy-provider.yaml API definition [proxy-provider:1.0.0]

Create an API using your default template
  $ apic config:set --global template-path="/etc/templates"
  $ ls /etc/templates
  proxy.hbs staging.hbs
  $ apic config:set --global template-default-api=staging
  $ apic create:api --title "Staging Provider"
  Created staging-provider.yaml API definition [staging-provider:1.0.0]
```

Options

```
      --api_type string        The type of api (rest, wsdl-to-rest, or wsdl) (default "wsdl")
      --basepath string        basepath value (default derived from name)
      --disable_ws_security    Disable generation of WS-Security definitions in api
      --filename string        filename (default derived from name)
      --gateway-type string    The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway) (default
"datapower-gateway")
  -h, --help                   Help for create:api
      --hostname string        host value (default $(catalog.host)
  -i, --interactive            use interactive mode
      --name string            x-ibm-name value (default derived from title)
      --product string         generate a product definition referencing the API
      --schemes string         list of schemes (valid options are http, https, ws and wss)
      --services string        service names separated by space
      --target-url string      target url
      --template string        use a provider template (if empty defaults to apic template)
      --title string           title value (required)
  -v, --version string         version value (default "1.0.0")
      --wsdl string            wsdl file to use as the source (required authentication via apic login)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic create:product

Create a product definition

Synopsis

Create a product definition

```
apic create:product [flags]
```

Examples

```
Create a product
  $ apic create:product --title "Climb On"
  Created climb-on.yaml product definition [climb-on:1.0.0]
Create a product interactively
  $ apic create:product
  ? Title: Climb On
  ? Name: climb-on
  ? File: climb-on.yaml
  ? Template:
  ? API Files:
  Created climb-on.yaml product definition [climb-on:1.0.0]
Create a product referencing existing APIs
  $ apic create:product --title "Climb On" --apis "routes.yaml ascents.yaml"
  Created climb-on.yaml product definition [climb-on:1.0.0]
Create an product using one of your templates
  $ apic config:set --global template-path="/etc/templates"
  $ ls /etc/templates
  proxy-product.hbs staging-product.hbs
  $ apic create:product --title "Proxy Product" --template proxy
  Created proxy-product.yaml product definition [proxy-product:1.0.0]
Create a product using your default template
  $ apic config:set --global template-path="/etc/templates"
  $ ls /etc/templates
  proxy-product.hbs staging-product.hbs
  $ apic config:set --global template-default-product=staging
```

```
  $ apic create:product --title "Staging Product"
  Created staging-product.yaml product definition [staging-product:1.0.0]
```

Options

```
      --apis string            api file names separated by a space
      --filename string        filename (default derived from name)
      --gateway-type string    The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway) (default
"datapower-gateway")
  -h, --help                   Help for create:product
  -i, --interactive            use interactive mode
      --name string            x-ibm-name value (default derived from title)
      --template string        use a provider template (if empty defaults to apic template)
      --title string           title value (required)
  -v, --version string         version value (default "1.0.0")
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials

Credentials operations

Synopsis

Credentials operations

```
apic credentials [flags]
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for credentials
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:clear

Credentials clear operations

Synopsis

Credentials clear operations

```
apic credentials:clear [flags]
```

Possible Usages

```
    apic credentials:clear [flags] --space-initiated
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --confirm string         Confirmation for critical updates (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for credentials:clear
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:create

Create a Application Credential object

Synopsis

Create a Application Credential object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
      - app_url
```

```
apic credentials:create [flags]
```

Possible Usages

```
      apic credentials:create [flags] --space-initiated CREDENTIAL_FILE
      apic credentials:create [flags] CREDENTIAL_FILE
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:delete

Delete the Application Credential object by name or id

Synopsis

Delete the Application Credential object by name or id

```
apic credentials:delete [flags]
```

Possible Usages

```
      apic credentials:delete [flags] --space-initiated CREDENTIAL
      apic credentials:delete [flags] CREDENTIAL
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:get

Get the Application Credential object by name or id

Synopsis

Get the Application Credential object by name or id

```
apic credentials:get [flags]
```

Possible Usages

```
      apic credentials:get [flags] --space-initiated CREDENTIAL
      apic credentials:get [flags] CREDENTIAL
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:list

List the Application Credential objects

Synopsis

List the Application Credential objects

```
apic credentials:list [flags]
```

Possible Usages

```
      apic credentials:list [flags] --space-initiated
      apic credentials:list [flags] --scope space
      apic credentials:list [flags] --scope catalog
      apic credentials:list [flags]
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:reset-client-secret

Reset the client secret

Synopsis

Reset the client secret

```
apic credentials:reset-client-secret [flags]
```

Possible Usages

```
    apic credentials:reset-client-secret [flags] --space-initiated CREDENTIAL
    apic credentials:reset-client-secret [flags] CREDENTIAL
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic credentials:reset

Reset the client id and client secret

Synopsis

Reset the client id and client secret

```
apic credentials:reset [flags]
```

Possible Usages

```
    apic credentials:reset [flags] --space-initiated CREDENTIAL
    apic credentials:reset [flags] CREDENTIAL
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
```

```
      --consumer-org string   Consumer Organization name or id (required)
      --format string         Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string            Organization name or id (required)
      --output string         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic credentials:update

Update the Application Credential object by name or id

Synopsis

Update the Application Credential object by name or id

```
      Fields not allowed to be null:
      - client_id

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
      - app_url
```

```
apic credentials:update [flags]
```

Possible Usages

```
      apic credentials:update [flags] --space-initiated CREDENTIAL CREDENTIAL_FILE
      apic credentials:update [flags] CREDENTIAL CREDENTIAL_FILE
```

Options

```
  -a, --app string            Application name or id (required)
  -c, --catalog string        Catalog name or id (required)
      --consumer-org string   Consumer Organization name or id (required)
      --format string         Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string            Organization name or id (required)
      --output string         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic credentials:verify-client-secret

Verify the client secret

Synopsis

Verify the client secret

```
apic credentials:verify-client-secret [flags]
```

Possible Usages

```
      apic credentials:verify-client-secret [flags] --space-initiated CREDENTIAL CREDENTIAL_FILE
      apic credentials:verify-client-secret [flags] CREDENTIAL CREDENTIAL_FILE
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic draft-apis

Draft Apis operations

Synopsis

Draft Apis operations

```
apic draft-apis [flags]
```

Options

```
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for draft-apis
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic draft-apis:clear-all

Clear all Draft API objects in all collections

Synopsis

Clear all Draft API objects in all collections

```
apic draft-apis:clear-all [flags]
```

Possible Usages

```
      apic draft-apis:clear-all [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:clear

Clear the Draft API objects

Synopsis

Clear the Draft API objects

```
apic draft-apis:clear [flags]
```

Possible Usages

```
      apic draft-apis:clear [flags] DRAFT_API_NAME
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:clone

Clone the draft-api objects

Synopsis

Clone the draft-api objects

```
apic draft-apis:clone [flags]
```

Possible Usages

```
      apic draft-apis:clone [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:create

Create a Draft API object

Synopsis

Create a Draft API object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic draft-apis:create [flags]
```

Possible Usages

```
      apic draft-apis:create [flags] DRAFT_API_FILE
```

Options

```
      --api_type string         The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --assembly_type string    The type of the assembly to generate (rest_to_proxy)
      --disable_ws_security     Disable generation of WS-Security definitions in api
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --gateway_type string     The type of the gateway (datapower-gateway, datapower-api-gateway, event-gateway)
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --wsdl_service string     Name of WSDL service to create the OpenAPI definition from
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:delete

Delete a Draft API

Synopsis

Delete a Draft API

```
apic draft-apis:delete [flags]
```

Possible Usages

```
      apic draft-apis:delete [flags] --id ID
      apic draft-apis:delete [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:document

Get the Draft API document by name and version

Synopsis

Get the Draft API document by name and version

```
apic draft-apis:document [flags]
```

Possible Usages

```
      apic draft-apis:document [flags] --id ID
      apic draft-apis:document [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:get

Get the Draft API object by name and version

Synopsis

Get the Draft API object by name and version

```
apic draft-apis:get [flags]
```

Possible Usages

```
      apic draft-apis:get [flags] --id ID
      apic draft-apis:get [flags] NAME:VERSION
```

Options

```
      --fields string   List of field names to return (default "add(wsdl,draft_api)")
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:list

List the Draft API objects

Synopsis

List the Draft API objects

```
apic draft-apis:list [flags]
```

Possible Usages

```
      apic draft-apis:list [flags] DRAFT_API_NAME
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:list-all

List all Draft API objects in all collections

Synopsis

List all Draft API objects in all collections

```
apic draft-apis:list-all [flags]
```

Possible Usages

```
      apic draft-apis:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:update

Update the Draft API object by name and version

Synopsis

Update the Draft API object by name and version

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic draft-apis:update [flags]
```

Possible Usages

```
      apic draft-apis:update [flags] --id ID DRAFT_API_FILE
      apic draft-apis:update [flags] NAME:VERSION DRAFT_API_FILE
```

Options

```
      --api_type string       The type of api (asyncapi, rest, graphql, wsdl_to_rest, or wsdl)
      --disable_ws_security   Disable generation of WS-Security definitions in api
      --format string         Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string            Organization name or id (required)
      --output string         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string         management server endpoint (required)
      --wsdl_service string   Name of WSDL service to create the OpenAPI definition from
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:update-wsdl

Update with a new wsdl

Synopsis

Update with a new wsdl

```
apic draft-apis:update-wsdl [flags]
```

Possible Usages

```
      apic draft-apis:update-wsdl [flags] --id ID DRAFT_APIS_FILE
      apic draft-apis:update-wsdl [flags] NAME:VERSION DRAFT_APIS_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:validate

Validate the draft api

Synopsis

Validate the draft api

```
apic draft-apis:validate [flags]
```

Possible Usages

```
      apic draft-apis:validate [flags] --id ID
      apic draft-apis:validate [flags] NAME:VERSION
```

Options

```
      --catalog_name string             Name of a catalog
      --format string                   Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway_service_names string    Names of Gateway Services
  -o, --org string                      Organization name or id (required)
      --output string                   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                    scope
  -s, --server string                   management server endpoint (required)
      --space_name string               Name of a space
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-apis:wsdl

Get the Draft API wsdl document by name and version

Synopsis

Get the Draft API wsdl document by name and version

```
apic draft-apis:wsdl [flags]
```

Possible Usages

```
      apic draft-apis:wsdl [flags] --id ID
      apic draft-apis:wsdl [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products

Draft Products operations

Synopsis

Draft Products operations

```
apic draft-products [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for draft-products
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:clear-all

Clear all Draft Product objects in all collections

Synopsis

Clear all Draft Product objects in all collections

```
apic draft-products:clear-all [flags]
```

Possible Usages

```
apic draft-products:clear-all [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:clear

Clear the Draft Product objects

Synopsis

Clear the Draft Product objects

```
apic draft-products:clear [flags]
```

Possible Usages

```
apic draft-products:clear [flags] DRAFT_PRODUCT_NAME
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:clone

Clone the draft-product objects

Synopsis

Clone the draft-product objects

```
apic draft-products:clone [flags]
```

Possible Usages

```
apic draft-products:clone [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
```

```
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:create

Create a Draft Product object

Synopsis

Create a Draft Product object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic draft-products:create [flags]
```

Possible Usages

```
      apic draft-products:create [flags] DRAFT_PRODUCT_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --product-only    Upload only the product document
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:delete

Delete a Draft Product

Synopsis

Delete a Draft Product

```
apic draft-products:delete [flags]
```

Possible Usages

```
      apic draft-products:delete [flags] --id ID
      apic draft-products:delete [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:document

Get the Draft Product document by name and version

Synopsis

Get the Draft Product document by name and version

```
apic draft-products:document [flags]
```

Possible Usages

```
      apic draft-products:document [flags] --id ID
      apic draft-products:document [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:get

Get the Draft Product object by name and version

Synopsis

Get the Draft Product object by name and version

```
apic draft-products:get [flags]
```

Possible Usages

```
      apic draft-products:get [flags] --id ID
      apic draft-products:get [flags] NAME:VERSION
```

Options

```
      --fields string   List of field names to return (default "add(draft_product,url)")
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:list-all

List all Draft Product objects in all collections

Synopsis

List all Draft Product objects in all collections

```
apic draft-products:list-all [flags]
```

Possible Usages

```
apic draft-products:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:list

List the Draft Product objects

Synopsis

List the Draft Product objects

```
apic draft-products:list [flags]
```

Possible Usages

```
apic draft-products:list [flags] DRAFT_PRODUCT_NAME
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic draft-products:publish

Publish a draft product by url reference

Synopsis

Publish a draft product by url reference

```
apic draft-products:publish [flags]
```

Possible Usages

```
apic draft-products:publish [flags] --stage --scope space DRAFT_PRODUCT_PUBLISH_FILE
apic draft-products:publish [flags] --stage DRAFT_PRODUCT_PUBLISH_FILE
```

```
      apic draft-products:publish [flags] --scope space DRAFT_PRODUCT_PUBLISH_FILE
      apic draft-products:publish [flags] DRAFT_PRODUCT_PUBLISH_FILE
```

Options

```
  -c, --catalog string             Catalog name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
      --gateway_services string    The list of gateway service names to support partial publishing
      --migrate_subscriptions      Migrate subscription when republish product
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic draft-products:update

Update the Draft Product object by name and version

Synopsis

Update the Draft Product object by name and version

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic draft-products:update [flags]
```

Possible Usages

```
      apic draft-products:update [flags] --id ID DRAFT_PRODUCT_FILE
      apic draft-products:update [flags] NAME:VERSION DRAFT_PRODUCT_FILE
```

Options

```
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic draft-products:validate

Validate the draft product

Synopsis

Validate the draft product

```
apic draft-products:validate [flags]
```

Possible Usages

```
      apic draft-products:validate [flags] --id ID
      apic draft-products:validate [flags] NAME:VERSION
```

Options

```
      --catalog_name string            Name of a catalog
      --format string                  Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway_service_names string   Names of Gateway Services
  -o, --org string                     Organization name or id (required)
      --output string                  Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                   scope
  -s, --server string                  management server endpoint (required)
      --space_name string              Name of a space
      --validate_apis                  Whether to validate APIs also
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drafts

Drafts operations

Synopsis

Drafts operations

```
apic drafts [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for drafts
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drafts:clear

Clear the Draft objects

Synopsis

Clear the Draft objects

```
apic drafts:clear [flags]
```

Possible Usages

```
    apic drafts:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drafts:list

List the Draft objects

Synopsis

List the Draft objects

```
apic drafts:list [flags]
```

Possible Usages

```
      apic drafts:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries

Entries operations

Synopsis

Entries operations

```
apic entries [flags]
```

Options

```
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries
      --limit int32         Maximum number of items to return
      --offset int32        Offset item number from list to begin return
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --truststore string   Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries:clear

Entries clear operations

Synopsis

Entries clear operations

```
apic entries:clear [flags]
```

Possible Usages

```
      apic entries:clear [flags] --scope truststore
```

Options

```
      --confirm string      Confirmation for critical updates (required)
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries:clear
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --truststore string   Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries:create

Entries create operations

Synopsis

Entries create operations

```
apic entries:create [flags]
```

Possible Usages

```
      apic entries:create [flags] --scope truststore ENTRY_FILE
```

Options

```
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries:create
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --truststore string   Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries:delete

Entries delete operations

Synopsis

Entries delete operations

```
apic entries:delete [flags]
```

Possible Usages

```
      apic entries:delete [flags] --scope truststore ENTRY
```

Options

```
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries:delete
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --truststore string   Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries:get

Entries get operations

Synopsis

Entries get operations

```
apic entries:get [flags]
```

Possible Usages

```
      apic entries:get [flags] --scope truststore ENTRY
```

Options

```
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries:get
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --truststore string   Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entries:list

Entries list operations

Synopsis

Entries list operations

```
apic entries:list [flags]
```

Possible Usages

```
      apic entries:list [flags] --scope truststore
```

Options

```
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                Help for entries:list
```

```
      --limit int32          Maximum number of items to return
      --offset int32         Offset item number from list to begin return
  -o, --org string           Organization name or id (required)
      --output string        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string         scope
  -s, --server string        management server endpoint (required)
      --truststore string    Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic entries:update

Entries update operations

Synopsis

Entries update operations

```
apic entries:update [flags]
```

Possible Usages

```
      apic entries:update [flags] --scope truststore ENTRY ENTRY_FILE
```

Options

```
      --format string        Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                 Help for entries:update
  -o, --org string           Organization name or id (required)
      --output string        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string         scope
  -s, --server string        management server endpoint (required)
      --truststore string    Truststore name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic extensions

Extensions operations

Synopsis

Extensions operations

```
apic extensions [flags]
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:clear-all

Extensions clear-all operations

Synopsis

Extensions clear-all operations

```
apic extensions:clear-all [flags]
```

Possible Usages

```
      apic extensions:clear-all [flags] --scope space
      apic extensions:clear-all [flags] --scope catalog
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --confirm string                       Confirmation for critical updates (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for extensions:clear-all
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:clear

Extensions clear operations

Synopsis

Extensions clear operations

```
apic extensions:clear [flags]
```

Possible Usages

```
      apic extensions:clear [flags] --scope space EXTENSION_NAME
      apic extensions:clear [flags] --scope catalog EXTENSION_NAME
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --confirm string                       Confirmation for critical updates (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for extensions:clear
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:clone

Extensions clone operations

Synopsis

Extensions clone operations

```
apic extensions:clone [flags]
```

Possible Usages

```
      apic extensions:clone [flags] --scope space
      apic extensions:clone [flags] --scope org
      apic extensions:clone [flags] --scope catalog
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions:clone
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:create

Extensions create operations

Synopsis

Extensions create operations

```
apic extensions:create [flags]
```

Possible Usages

```
      apic extensions:create [flags] --scope space EXTENSION_FILE
      apic extensions:create [flags] --scope catalog EXTENSION_FILE
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for extensions:create
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:delete

Extensions delete operations

Synopsis

Extensions delete operations

```
apic extensions:delete [flags]
```

Possible Usages

```
      apic extensions:delete [flags] --id --scope space ID
      apic extensions:delete [flags] --id --scope catalog ID
      apic extensions:delete [flags] --scope space NAME:VERSION
      apic extensions:delete [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for extensions:delete
      --id                                  id
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:document

Extensions document operations

Synopsis

Extensions document operations

```
apic extensions:document [flags]
```

Possible Usages

```
      apic extensions:document [flags] --id --scope space ID
      apic extensions:document [flags] --id --scope org ID
      apic extensions:document [flags] --id --scope catalog ID
      apic extensions:document [flags] --scope space NAME:VERSION
      apic extensions:document [flags] --scope org NAME:VERSION
      apic extensions:document [flags] --scope catalog NAME:VERSION
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions:document
      --id                                  id
```

```
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:get

Extensions get operations

Synopsis

Extensions get operations

```
apic extensions:get [flags]
```

Possible Usages

```
      apic extensions:get [flags] --id --scope space ID
      apic extensions:get [flags] --id --scope org ID
      apic extensions:get [flags] --id --scope catalog ID
      apic extensions:get [flags] --scope space NAME:VERSION
      apic extensions:get [flags] --scope org NAME:VERSION
      apic extensions:get [flags] --scope catalog NAME:VERSION
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions:get
      --id                                  id
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:list

Extensions list operations

Synopsis

Extensions list operations

```
apic extensions:list [flags]
```

Possible Usages

```
      apic extensions:list [flags] --scope space EXTENSION_NAME
      apic extensions:list [flags] --scope org EXTENSION_NAME
      apic extensions:list [flags] --scope catalog EXTENSION_NAME
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions:list
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:list-all

Extensions list-all operations

Synopsis

Extensions list-all operations

```
apic extensions:list-all [flags]
```

Possible Usages

```
     apic extensions:list-all [flags] --scope space
     apic extensions:list-all [flags] --scope org
     apic extensions:list-all [flags] --scope catalog
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for extensions:list-all
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic extensions:update

Extensions update operations

Synopsis

Extensions update operations

```
apic extensions:update [flags]
```

Possible Usages

```
      apic extensions:update [flags] --id --scope space ID EXTENSION_FILE
      apic extensions:update [flags] --id --scope catalog ID EXTENSION_FILE
      apic extensions:update [flags] --scope space NAME:VERSION EXTENSION_FILE
      apic extensions:update [flags] --scope catalog NAME:VERSION EXTENSION_FILE
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for extensions:update
      --id                                   id
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-extensions

Gateway Extensions operations

Synopsis

Gateway Extensions operations

```
apic gateway-extensions [flags]
```

Options

```
  -h, --help   Help for gateway-extensions
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-extensions:create

Gateway Extensions create operations

Synopsis

Gateway Extensions create operations

```
apic gateway-extensions:create [flags]
```

Possible Usages

```
      apic gateway-extensions:create [flags] --scope org GATEWAY_EXTENSIONS_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway-service string     Gateway Service name or id (required)
  -h, --help                       Help for gateway-extensions:create
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-extensions:delete

Gateway Extensions delete operations

Synopsis

Gateway Extensions delete operations

```
apic gateway-extensions:delete [flags]
```

Possible Usages

```
      apic gateway-extensions:delete [flags] --scope org
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway-service string     Gateway Service name or id (required)
  -h, --help                       Help for gateway-extensions:delete
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-extensions:get

Gateway Extensions get operations

Synopsis

Gateway Extensions get operations

```
apic gateway-extensions:get [flags]
```

Possible Usages

```
      apic gateway-extensions:get [flags] --scope org
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway-service string     Gateway Service name or id (required)
  -h, --help                       Help for gateway-extensions:get
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-extensions:implementation

Get the Gateway Extension implementation document

Synopsis

Get the Gateway Extension implementation document

```
apic gateway-extensions:implementation [flags]
```

Possible Usages

```
      apic gateway-extensions:implementation [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway-service string     Gateway Service name or id (required)
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-extensions:update

Gateway Extensions update operations

Synopsis

Gateway Extensions update operations

```
apic gateway-extensions:update [flags]
```

Possible Usages

```
      apic gateway-extensions:update [flags] --scope org GATEWAY_EXTENSION_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --gateway-service string     Gateway Service name or id (required)
  -h, --help                       Help for gateway-extensions:update
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-services

Gateway Services operations

Synopsis

Gateway Services operations

```
apic gateway-services [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                       Help for gateway-services
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-services:clear

Clear the Gateway Service objects

Synopsis

Clear the Gateway Service objects

```
apic gateway-services:clear [flags]
```

Possible Usages

```
      apic gateway-services:clear [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --confirm string             Confirmation for critical updates (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-services:create

Create a Gateway Service object

Synopsis

Create a Gateway Service object

```
      Required fields:
      - endpoint
      - api_endpoint_base
      - sni

      Fields not allowed:
      - owned
```

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic gateway-services:create [flags]
```

Possible Usages

```
      apic gateway-services:create [flags] GATEWAY_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-services:delete

Delete the Gateway Service object by name or id

Synopsis

Delete the Gateway Service object by name or id

```
apic gateway-services:delete [flags]
```

Possible Usages

```
      apic gateway-services:delete [flags] GATEWAY_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-services:get

Get the Gateway Service object by name or id

Synopsis

Get the Gateway Service object by name or id

```
apic gateway-services:get [flags]
```

Possible Usages

```
      apic gateway-services:get [flags] --scope org GATEWAY_SERVICE
      apic gateway-services:get [flags] GATEWAY_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-services:list

List the Gateway Service objects

Synopsis

List the Gateway Service objects

```
apic gateway-services:list [flags]
```

Possible Usages

```
      apic gateway-services:list [flags] --scope org
      apic gateway-services:list [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic gateway-services:reset-oauth-secret

Reset the oauth shared crypto material

Synopsis

Reset the oauth shared crypto material

```
apic gateway-services:reset-oauth-secret [flags]
```

Possible Usages

```
      apic gateway-services:reset-oauth-secret [flags] GATEWAY_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic gateway-services:update

Update the Gateway Service object by name or id

Synopsis

Update the Gateway Service object by name or id

```
      Fields not allowed to be null:
      - endpoint
      - api_endpoint_base
      - sni

      Fields not allowed:
      - owned
      - endpoint
      - integration_url
      - gateway_service_type
      - api_endpoint_base

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic gateway-services:update [flags]
```

Possible Usages

```
      apic gateway-services:update [flags] GATEWAY_SERVICE GATEWAY_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic global-policies

Global Policies operations

Synopsis

Global Policies operations

```
apic global-policies [flags]
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --fields string                         List of field names to return
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies
      --limit int32                           Maximum number of items to return
      --offset int32                          Offset item number from list to begin return
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:clear

Global Policies clear operations

Synopsis

Global Policies clear operations

```
apic global-policies:clear [flags]
```

Possible Usages

```
    apic global-policies:clear [flags] --scope space GLOBAL_POLICY_NAME
    apic global-policies:clear [flags] --scope catalog GLOBAL_POLICY_NAME
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --confirm string                        Confirmation for critical updates (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:clear
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:clear-all

Global Policies clear-all operations

Synopsis

Global Policies clear-all operations

```
apic global-policies:clear-all [flags]
```

Possible Usages

```
    apic global-policies:clear-all [flags] --scope space
    apic global-policies:clear-all [flags] --scope catalog
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --confirm string                        Confirmation for critical updates (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:clear-all
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:create

Global Policies create operations

Synopsis

Global Policies create operations

```
apic global-policies:create [flags]
```

Possible Usages

```
      apic global-policies:create [flags] --scope space GLOBAL_POLICY_FILE
      apic global-policies:create [flags] --scope catalog GLOBAL_POLICY_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:create
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:delete

Global Policies delete operations

Synopsis

Global Policies delete operations

```
apic global-policies:delete [flags]
```

Possible Usages

```
      apic global-policies:delete [flags] --id --scope space ID
      apic global-policies:delete [flags] --id --scope catalog ID
      apic global-policies:delete [flags] --scope space NAME:VERSION
      apic global-policies:delete [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:delete
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:document

Global Policies document operations

Synopsis

Global Policies document operations

```
apic global-policies:document [flags]
```

Possible Usages

```
      apic global-policies:document [flags] --id --scope space ID
      apic global-policies:document [flags] --id --scope catalog ID
      apic global-policies:document [flags] --scope space NAME:VERSION
      apic global-policies:document [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:document
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:get

Global Policies get operations

Synopsis

Global Policies get operations

```
apic global-policies:get [flags]
```

Possible Usages

```
      apic global-policies:get [flags] --id --scope space ID
      apic global-policies:get [flags] --id --scope catalog ID
```

```
      apic global-policies:get [flags] --scope space NAME:VERSION
      apic global-policies:get [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policies:get
      --id                                   id
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:list

Global Policies list operations

Synopsis

Global Policies list operations

```
apic global-policies:list [flags]
```

Possible Usages

```
      apic global-policies:list [flags] --scope space GLOBAL_POLICY_NAME
      apic global-policies:list [flags] --scope catalog GLOBAL_POLICY_NAME
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policies:list
      --limit int32                          Maximum number of items to return
      --offset int32                         Offset item number from list to begin return
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:list-all

Global Policies list-all operations

Synopsis

Global Policies list-all operations

```
apic global-policies:list-all [flags]
```

Possible Usages

```
      apic global-policies:list-all [flags] --scope space
      apic global-policies:list-all [flags] --scope catalog
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --fields string                         List of field names to return
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:list-all
      --limit int32                           Maximum number of items to return
      --offset int32                          Offset item number from list to begin return
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policies:update

Global Policies update operations

Synopsis

Global Policies update operations

```
apic global-policies:update [flags]
```

Possible Usages

```
      apic global-policies:update [flags] --id --scope space ID GLOBAL_POLICY_FILE
      apic global-policies:update [flags] --id --scope catalog ID GLOBAL_POLICY_FILE
      apic global-policies:update [flags] --scope space NAME:VERSION GLOBAL_POLICY_FILE
      apic global-policies:update [flags] --scope catalog NAME:VERSION GLOBAL_POLICY_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policies:update
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-errors

Global Policy Errors operations

Synopsis

Global Policy Errors operations

```
apic global-policy-errors [flags]
```

Options

```
  -h, --help   Help for global-policy-errors
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-errors:create

Global Policy Errors create operations

Synopsis

Global Policy Errors create operations

```
apic global-policy-errors:create [flags]
```

Possible Usages

```
      apic global-policy-errors:create [flags] --scope space GLOBAL_POLICY_ERROR_FILE
      apic global-policy-errors:create [flags] --scope catalog GLOBAL_POLICY_ERROR_FILE
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policy-errors:create
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-errors:delete

Global Policy Errors delete operations

Synopsis

Global Policy Errors delete operations

```
apic global-policy-errors:delete [flags]
```

Possible Usages

```
      apic global-policy-errors:delete [flags] --scope space
      apic global-policy-errors:delete [flags] --scope catalog
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policy-errors:delete
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-errors:get

Global Policy Errors get operations

Synopsis

Global Policy Errors get operations

```
apic global-policy-errors:get [flags]
```

Possible Usages

```
      apic global-policy-errors:get [flags] --scope space
      apic global-policy-errors:get [flags] --scope catalog
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-errors:get
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-errors:update

Global Policy Errors update operations

Synopsis

Global Policy Errors update operations

```
apic global-policy-errors:update [flags]
```

Possible Usages

```
      apic global-policy-errors:update [flags] --scope space GLOBAL_POLICY_ERROR_FILE
      apic global-policy-errors:update [flags] --scope catalog GLOBAL_POLICY_ERROR_FILE
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-errors:update
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-posthooks

Global Policy Posthooks operations

Synopsis

Global Policy Posthooks operations

```
apic global-policy-posthooks [flags]
```

Options

```
  -h, --help   Help for global-policy-posthooks
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-posthooks:create

Global Policy Posthooks create operations

Synopsis

Global Policy Posthooks create operations

```
apic global-policy-posthooks:create [flags]
```

Possible Usages

```
      apic global-policy-posthooks:create [flags] --scope space GLOBAL_POLICY_POSTHOOK_FILE
      apic global-policy-posthooks:create [flags] --scope catalog GLOBAL_POLICY_POSTHOOK_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-posthooks:create
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-posthooks:delete

Global Policy Posthooks delete operations

Synopsis

Global Policy Posthooks delete operations

```
apic global-policy-posthooks:delete [flags]
```

Possible Usages

```
      apic global-policy-posthooks:delete [flags] --scope space
      apic global-policy-posthooks:delete [flags] --scope catalog
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policy-posthooks:delete
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-posthooks:get

Global Policy Posthooks get operations

Synopsis

Global Policy Posthooks get operations

```
apic global-policy-posthooks:get [flags]
```

Possible Usages

```
      apic global-policy-posthooks:get [flags] --scope space
      apic global-policy-posthooks:get [flags] --scope catalog
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --fields string                         List of field names to return
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for global-policy-posthooks:get
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-posthooks:update

Global Policy Posthooks update operations

Synopsis

Global Policy Posthooks update operations

```
apic global-policy-posthooks:update [flags]
```

Possible Usages

```
      apic global-policy-posthooks:update [flags] --scope space GLOBAL_POLICY_POSTHOOK_FILE
      apic global-policy-posthooks:update [flags] --scope catalog GLOBAL_POLICY_POSTHOOK_FILE
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policy-posthooks:update
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-prehooks

Global Policy Prehooks operations

Synopsis

Global Policy Prehooks operations

```
apic global-policy-prehooks [flags]
```

Options

```
  -h, --help   Help for global-policy-prehooks
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-prehooks:create

Global Policy Prehooks create operations

Synopsis

Global Policy Prehooks create operations

```
apic global-policy-prehooks:create [flags]
```

Possible Usages

```
      apic global-policy-prehooks:create [flags] --scope space GLOBAL_POLICY_PREHOOK_FILE
      apic global-policy-prehooks:create [flags] --scope catalog GLOBAL_POLICY_PREHOOK_FILE
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for global-policy-prehooks:create
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-prehooks:delete

Global Policy Prehooks delete operations

Synopsis

Global Policy Prehooks delete operations

```
apic global-policy-prehooks:delete [flags]
```

Possible Usages

```
      apic global-policy-prehooks:delete [flags] --scope space
      apic global-policy-prehooks:delete [flags] --scope catalog
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-prehooks:delete
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-prehooks:get

Global Policy Prehooks get operations

Synopsis

Global Policy Prehooks get operations

```
apic global-policy-prehooks:get [flags]
```

Possible Usages

```
      apic global-policy-prehooks:get [flags] --scope space
      apic global-policy-prehooks:get [flags] --scope catalog
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-prehooks:get
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic global-policy-prehooks:update

Global Policy Prehooks update operations

Synopsis

Global Policy Prehooks update operations

```
apic global-policy-prehooks:update [flags]
```

Possible Usages

```
      apic global-policy-prehooks:update [flags] --scope space GLOBAL_POLICY_PREHOOK_FILE
      apic global-policy-prehooks:update [flags] --scope catalog GLOBAL_POLICY_PREHOOK_FILE
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for global-policy-prehooks:update
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups

Groups operations

Synopsis

Groups operations

```
apic groups [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups
      --limit int32       Maximum number of items to return
      --offset int32      Offset item number from list to begin return
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:clear

Groups clear operations

Synopsis

Groups clear operations

```
apic groups:clear [flags]
```

Possible Usages

```
      apic groups:clear [flags] --space-initiated
      apic groups:clear [flags] --scope cloud
      apic groups:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --confirm string    Confirmation for critical updates (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:clear
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:create

Groups create operations

Synopsis

Groups create operations

```
apic groups:create [flags]
```

Possible Usages

```
      apic groups:create [flags] --space-initiated GROUP_FILE
      apic groups:create [flags] --scope cloud GROUP_FILE
      apic groups:create [flags] --scope catalog GROUP_FILE
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:create
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:delete

Groups delete operations

Synopsis

Groups delete operations

```
apic groups:delete [flags]
```

Possible Usages

```
      apic groups:delete [flags] --space-initiated GROUP
      apic groups:delete [flags] --scope cloud GROUP
      apic groups:delete [flags] --scope catalog GROUP
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:delete
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:get

Groups get operations

Synopsis

Groups get operations

```
apic groups:get [flags]
```

Possible Usages

```
      apic groups:get [flags] --space-initiated GROUP
      apic groups:get [flags] --scope cloud GROUP
      apic groups:get [flags] --scope catalog GROUP
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:get
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:list

Groups list operations

Synopsis

Groups list operations

```
apic groups:list [flags]
```

Possible Usages

```
      apic groups:list [flags] --space-initiated
      apic groups:list [flags] --scope cloud
      apic groups:list [flags] --scope catalog
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:list
      --limit int32       Maximum number of items to return
      --offset int32      Offset item number from list to begin return
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic groups:update

Groups update operations

Synopsis

Groups update operations

```
apic groups:update [flags]
```

Possible Usages

```
      apic groups:update [flags] --space-initiated GROUP GROUP_FILE
      apic groups:update [flags] --scope cloud GROUP GROUP_FILE
      apic groups:update [flags] --scope catalog GROUP GROUP_FILE
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for groups:update
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic iam-apikey

Log in to an IBM API Connect cloud Reserved Instance using an IBM Cloud API key

Synopsis

Log in to an IBM API Connect cloud Reserved Instance using an IBM Cloud API key

```
apic iam-apikey [flags]
```

Examples

```
Interactive iam-apikey login
$ apic iam-apikey
Enter your IBM Cloud details
? Server: mgmnthost.com
? IBM Cloud API Key?: my-apikey
 Logged into mgmnthost.com successfully

Non-interactive
$ apic iam-apikey --apiKey <apikey> --server mgmnthost.com
Logged into mgmnthost.com successfully
```

Options

```
      --apiKey string           IBM cloud api key
  -h, --help                    Help for iam-apikey
  -s, --server string           management server endpoint
      --tokenendpoint string    iam server endpoint (default: iam.cloud.ibm.com)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic identity-providers

Identity Providers operations

Synopsis

Identity Providers operations

```
apic identity-providers [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for identity-providers
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string    scope
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic identity-providers:list

Identity Providers list operations

Synopsis

Identity Providers list operations

```
apic identity-providers:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for identity-providers:list
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string    scope
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic integrations

Integrations operations

Synopsis

Integrations operations

```
apic integrations [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for integrations
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic integrations:clear

Integrations clear operations

Synopsis

Integrations clear operations

```
apic integrations:clear [flags]
```

Options

```
      --confirm string          Confirmation for critical updates (required)
      --format string           Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                    Help for integrations:clear
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic integrations:create

Integrations create operations

Synopsis

Integrations create operations

```
apic integrations:create [flags]
```

Options

```
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for integrations:create
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic integrations:delete

Integrations delete operations

Synopsis

Integrations delete operations

```
apic integrations:delete [flags]
```

Options

```
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for integrations:delete
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic integrations:get

Integrations get operations

Synopsis

Integrations get operations

```
apic integrations:get [flags]
```

Options

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for integrations:get
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
```

```
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic integrations:list

Integrations list operations

Synopsis

Integrations list operations

```
apic integrations:list [flags]
```

Options

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for integrations:list
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic integrations:list-all

List all Integration objects in all collections

Synopsis

List all Integration objects in all collections

```
apic integrations:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for integrations:list-all
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic integrations:update

Integrations update operations

Synopsis

Integrations update operations

```
apic integrations:update [flags]
```

Options

```
      --format string           Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                    Help for integrations:update
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic invitations

Invitations operations

Synopsis

Invitations operations

```
apic invitations [flags]
```

Options

```
  -c, --catalog string      Catalog name or id (required)
      --fields string       List of field names to return
      --format string       Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                Help for invitations
      --limit int32         Maximum number of items to return
      --offset int32        Offset item number from list to begin return
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
      --space string        Space name or id (required)
      --space-initiated     space-initiated
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic invitations:clear

Invitations clear operations

Synopsis

Invitations clear operations

```
apic invitations:clear [flags]
```

Options

```
  -c, --catalog string      Catalog name or id (required)
      --confirm string      Confirmation for critical updates (required)
      --format string       Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                Help for invitations:clear
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string        scope
  -s, --server string       management server endpoint (required)
```

```
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic invitations:create

Invitations create operations

Synopsis

Invitations create operations

```
apic invitations:create [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for invitations:create
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic invitations:delete

Invitations delete operations

Synopsis

Invitations delete operations

```
apic invitations:delete [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for invitations:delete
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic invitations:get

Invitations get operations

Synopsis

Invitations get operations

```
apic invitations:get [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for invitations:get
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic invitations:list

Invitations list operations

Synopsis

Invitations list operations

```
apic invitations:list [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for invitations:list
      --limit int32       Maximum number of items to return
      --offset int32      Offset item number from list to begin return
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic invitations:update

Invitations update operations

Synopsis

Invitations update operations

```
apic invitations:update [flags]
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for invitations:update
  -o, --org string        Organization name or id (required)
```

```
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
      --space string      Space name or id (required)
      --space-initiated   space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic jobs

Jobs operations

Synopsis

Jobs operations

```
apic jobs [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic jobs:clear

Clear the Job objects

Synopsis

Clear the Job objects

```
apic jobs:clear [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic jobs:delete

Delete the Job object by name or id

Synopsis

Delete the Job object by name or id

```
apic jobs:delete [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs:delete
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic jobs:get

Get the Job object by name or id

Synopsis

Get the Job object by name or id

```
apic jobs:get [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic jobs:list

List the Job objects

Synopsis

List the Job objects

```
apic jobs:list [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic jobs:retry

Re-attempt blocked or failed job

Synopsis

Re-attempt blocked or failed job

```
apic jobs:retry [flags]
```

Options

```
      --billing string   Billing name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for jobs:retry
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic keystores

Keystores operations

Synopsis

Keystores operations

```
apic keystores [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic keystores:clear

Clear the Keystore objects

Synopsis

Clear the Keystore objects

```
apic keystores:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for keystores:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic keystores:create

Create a Keystore object

Synopsis

Create a Keystore object Required fields:\ - keystore Fields allowed but ignored:\ - id - type - api_version - scope - created_at - updated_at - url

```
apic keystores:create [flags]
```

Options

```
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores:create
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic keystores:delete

Delete the Keystore object by name or id

Synopsis

Delete the Keystore object by name or id

```
apic keystores:delete [flags]
```

Options

```
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores:delete
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic keystores:get

Get the Keystore object by name or id

Synopsis

Get the Keystore object by name or id

```
apic keystores:get [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores:get
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic keystores:list

List the Keystore objects

Synopsis

List the Keystore objects

```
apic keystores:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores:list
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic keystores:update

Update the Keystore object by name or id

Synopsis

Update the Keystore object by name or id Fields allowed but ignored:\ - id - type - api_version - scope - created_at - updated_at - url

```
apic keystores:update [flags]
```

Options

```
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for keystores:update
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic licenses

Review the license for API Connect

Synopsis

Review the license for API Connect

```
apic licenses [flags]
```

Options

```
  -h, --help              Help for licenses
      --non-ibm-license   Display the non IBM license files.
      --notices           Display the notices file
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic log-spec

Log Spec operations

Synopsis

Log Spec operations

```
apic log-spec [flags]
```

Options

```
  -h, --help   Help for log-spec
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic log-spec:get

Get the Log Spec object

Synopsis

Get the Log Spec object

```
apic log-spec:get [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for log-spec:get
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic log-spec:update

Update the Log Spec object

Synopsis

Update the Log Spec object Fields allowed but ignored:\ - name - type - api_version - scope - created_at - updated_at - url

```
apic log-spec:update [flags]
```

Options

```
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for log-spec:update
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic login

Log in to an IBM API Connect cloud

Synopsis

Log in to an IBM API Connect cloud

```
apic login [flags]
```

Examples

```
Interactive login
$ apic login
Enter your API Connect credentials
? Server: mgmnthost.com
? Realm: company/realm
? Username: tommy
? Password: password
Logged into mgmnthost.com successfully

Non-interactive login
$ apic login --username tommy --password password --server mgmnthost.com --realm company/realm
Logged into mgmnthost.com successfully
```

Options

```
      --apiKey string     apiKey
      --context string    context
  -h, --help              Help for login
  -p, --password string   password
  -r, --realm string      realm
  -s, --server string     management server endpoint
      --sso               sso
  -u, --username string   user name
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic logout

Log out of an IBM API Connect cloud

Synopsis

Log out of an IBM API Connect cloud

```
apic logout [flags]
```

Examples

```
Clear the local authentication credentials for mgmnthost.com
$ apic logout --server mgmnthost.com
Logged out of server mgmnthost.com
```

Options

```
  -h, --help            Help for logout
  -s, --server string   management server endpoint
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic mail-servers

Mail Servers operations

Synopsis

Mail Servers operations

```
apic mail-servers [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for mail-servers
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic mail-servers:clear

Clear the Mail Server objects

Synopsis

Clear the Mail Server objects

```
apic mail-servers:clear [flags]
```

Possible Usages

```
    apic mail-servers:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic mail-servers:create

Create a Mail Server object

Synopsis

Create a Mail Server object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic mail-servers:create [flags]
```

Possible Usages

```
      apic mail-servers:create [flags] MAIL_SERVER_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic mail-servers:delete

Delete the Mail Server object by name or id

Synopsis

Delete the Mail Server object by name or id

```
apic mail-servers:delete [flags]
```

Possible Usages

```
      apic mail-servers:delete [flags] MAIL_SERVER
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic mail-servers:get

Get the Mail Server object by name or id

Synopsis

Get the Mail Server object by name or id

```
apic mail-servers:get [flags]
```

Possible Usages

```
      apic mail-servers:get [flags] MAIL_SERVER
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic mail-servers:list

List the Mail Server objects

Synopsis

List the Mail Server objects

```
apic mail-servers:list [flags]
```

Possible Usages

```
      apic mail-servers:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic mail-servers:test-connection

Test a Mail Server connection

Synopsis

Test a Mail Server connection

```
apic mail-servers:test-connection [flags]
```

Possible Usages

```
      apic mail-servers:test-connection [flags] --test-config-only MAIL_SERVER_PROVIDED_TEST_CONNECTION_FILE
      apic mail-servers:test-connection [flags] MAIL_SERVER MAIL_SERVER_SAVED_TEST_CONNECTION_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic mail-servers:update

Update the Mail Server object by name or id

Synopsis

Update the Mail Server object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic mail-servers:update [flags]
```

Possible Usages

```
      apic mail-servers:update [flags] MAIL_SERVER MAIL_SERVER_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic me

Me operations

Synopsis

Me operations

```
apic me [flags]
```

Options

```
  -h, --help   Help for me
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic me:change-password

Change my password

Synopsis

Change my password

```
apic me:change-password [flags]
```

Possible Usages

```
      apic me:change-password [flags] CHANGE_PASSWORD_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic me:delete

Delete the Me object

Synopsis

Delete the Me object

```
apic me:delete [flags]
```

Possible Usages

```
      apic me:delete [flags]
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic me:get

Get the Me object

Synopsis

Get the Me object

```
apic me:get [flags]
```

Possible Usages

```
      apic me:get [flags]
```

Options

```
      --context string   user's login context admin/manager
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic me:update

Update the Me object

Synopsis

Update the Me object

```
      Fields allowed but ignored:
      - name
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic me:update [flags]
```

Possible Usages

```
      apic me:update [flags] ME_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic member-invitations

Member Invitations operations

Synopsis

Member Invitations operations

```
apic member-invitations [flags]
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --consumer-org string     Consumer Organization name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for member-invitations
      --limit int32             Maximum number of items to return
      --offset int32            Offset item number from list to begin return
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --space-initiated         space-initiated
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic member-invitations:clear

Member Invitations clear operations

Synopsis

Member Invitations clear operations

```
apic member-invitations:clear [flags]
```

Possible Usages

```
    apic member-invitations:clear [flags] --scope consumer-org --space-initiated
    apic member-invitations:clear [flags] --scope space
    apic member-invitations:clear [flags] --scope org
    apic member-invitations:clear [flags] --scope consumer-org
    apic member-invitations:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --confirm string          Confirmation for critical updates (required)
      --consumer-org string     Consumer Organization name or id (required)
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for member-invitations:clear
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --space-initiated         space-initiated
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic member-invitations:create

Member Invitations create operations

Synopsis

Member Invitations create operations

```
apic member-invitations:create [flags]
```

Possible Usages

```
      apic member-invitations:create [flags] --scope consumer-org --space-initiated MEMBER_INVITATION_FILE
      apic member-invitations:create [flags] --scope space MEMBER_INVITATION_FILE
      apic member-invitations:create [flags] --scope org MEMBER_INVITATION_FILE
      apic member-invitations:create [flags] --scope consumer-org MEMBER_INVITATION_FILE
      apic member-invitations:create [flags] --scope catalog MEMBER_INVITATION_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for member-invitations:create
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic member-invitations:delete

Member Invitations delete operations

Synopsis

Member Invitations delete operations

```
apic member-invitations:delete [flags]
```

Possible Usages

```
      apic member-invitations:delete [flags] --scope consumer-org --space-initiated MEMBER_INVITATION
      apic member-invitations:delete [flags] --scope space MEMBER_INVITATION
      apic member-invitations:delete [flags] --scope org MEMBER_INVITATION
      apic member-invitations:delete [flags] --scope consumer-org MEMBER_INVITATION
      apic member-invitations:delete [flags] --scope catalog MEMBER_INVITATION
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for member-invitations:delete
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic member-invitations:get

Member Invitations get operations

Synopsis

Member Invitations get operations

```
apic member-invitations:get [flags]
```

Possible Usages

```
      apic member-invitations:get [flags] --scope consumer-org --space-initiated MEMBER_INVITATION
      apic member-invitations:get [flags] --scope space MEMBER_INVITATION
      apic member-invitations:get [flags] --scope org MEMBER_INVITATION
      apic member-invitations:get [flags] --scope consumer-org MEMBER_INVITATION
      apic member-invitations:get [flags] --scope catalog MEMBER_INVITATION
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for member-invitations:get
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic member-invitations:list

Member Invitations list operations

Synopsis

Member Invitations list operations

```
apic member-invitations:list [flags]
```

Possible Usages

```
      apic member-invitations:list [flags] --scope consumer-org --space-initiated
      apic member-invitations:list [flags] --scope space
      apic member-invitations:list [flags] --scope org
      apic member-invitations:list [flags] --scope consumer-org
      apic member-invitations:list [flags] --scope catalog
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for member-invitations:list
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic member-invitations:update

Member Invitations update operations

Synopsis

Member Invitations update operations

```
apic member-invitations:update [flags]
```

Possible Usages

```
      apic member-invitations:update [flags] --scope consumer-org --space-initiated MEMBER_INVITATION MEMBER_INVITATION_FILE
      apic member-invitations:update [flags] --scope space MEMBER_INVITATION MEMBER_INVITATION_FILE
      apic member-invitations:update [flags] --scope org MEMBER_INVITATION MEMBER_INVITATION_FILE
      apic member-invitations:update [flags] --scope consumer-org MEMBER_INVITATION MEMBER_INVITATION_FILE
      apic member-invitations:update [flags] --scope catalog MEMBER_INVITATION MEMBER_INVITATION_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for member-invitations:update
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members

Members operations

Synopsis

Members operations

```
apic members [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:clear

Members clear operations

Synopsis

Members clear operations

```
apic members:clear [flags]
```

Possible Usages

```
      apic members:clear [flags] --scope consumer-org --space-initiated
      apic members:clear [flags] --scope space
      apic members:clear [flags] --scope org
      apic members:clear [flags] --scope consumer-org
      apic members:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --confirm string         Confirmation for critical updates (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:clear
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:create

Members create operations

Synopsis

Members create operations

```
apic members:create [flags]
```

Possible Usages

```
      apic members:create [flags] --scope consumer-org --space-initiated MEMBER_FILE
      apic members:create [flags] --scope space MEMBER_FILE
      apic members:create [flags] --scope org MEMBER_FILE
      apic members:create [flags] --scope consumer-org MEMBER_FILE
      apic members:create [flags] --scope catalog MEMBER_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:create
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:delete

Members delete operations

Synopsis

Members delete operations

```
apic members:delete [flags]
```

Possible Usages

```
      apic members:delete [flags] --scope consumer-org --space-initiated MEMBER
      apic members:delete [flags] --scope space MEMBER
      apic members:delete [flags] --scope org MEMBER
      apic members:delete [flags] --scope consumer-org MEMBER
      apic members:delete [flags] --scope catalog MEMBER
```

Options

```
      --cascade                Cascade the behavior
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:delete
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:get

Members get operations

Synopsis

Members get operations

```
apic members:get [flags]
```

Possible Usages

```
      apic members:get [flags] --scope consumer-org --space-initiated MEMBER
      apic members:get [flags] --scope space MEMBER
      apic members:get [flags] --scope org MEMBER
      apic members:get [flags] --scope consumer-org MEMBER
      apic members:get [flags] --scope catalog MEMBER
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:get
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:list

Members list operations

Synopsis

Members list operations

```
apic members:list [flags]
```

Possible Usages

```
      apic members:list [flags] --scope consumer-org --space-initiated
      apic members:list [flags] --scope space
      apic members:list [flags] --scope org
      apic members:list [flags] --scope consumer-org
      apic members:list [flags] --scope catalog
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:list
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic members:update

Members update operations

Synopsis

Members update operations

```
apic members:update [flags]
```

Possible Usages

```
      apic members:update [flags] --scope consumer-org --space-initiated MEMBER MEMBER_FILE
      apic members:update [flags] --scope space MEMBER MEMBER_FILE
      apic members:update [flags] --scope org MEMBER MEMBER_FILE
      apic members:update [flags] --scope consumer-org MEMBER MEMBER_FILE
      apic members:update [flags] --scope catalog MEMBER MEMBER_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for members:update
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic monitor

Monitor your APIs

Synopsis

Run your API Tests repeatedly to identify issues.

```
apic monitor [flags]
```

Options

```
  -h, --help   Help for monitor
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-languages

Notification Languages operations

Synopsis

Notification Languages operations

```
apic notification-languages [flags]
```

Options

```
  -c, --catalog string                Catalog name or id (required)
      --format string                 Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                          Help for notification-languages
      --notification-template string  Notification Template name or id (required)
  -o, --org string                    Organization name or id (required)
      --output string                 Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                  scope
  -s, --server string                 management server endpoint (required)
      --space string                  Space name or id (required)
      --subcollection string          subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-languages:get

Notification Languages get operations

Synopsis

Notification Languages get operations

```
apic notification-languages:get [flags]
```

Possible Usages

```
      apic notification-languages:get [flags] --scope space --subcollection space NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope space --subcollection consumer NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope org --subcollection space NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope org --subcollection provider NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope org --subcollection consumer NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope org --subcollection catalog NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope org --subcollection admin NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection space NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection provider NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection consumer NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection cloud NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection catalog NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope cloud --subcollection admin NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope catalog --subcollection space NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope catalog --subcollection consumer NOTIFICATION_LANGUAGE
      apic notification-languages:get [flags] --scope catalog --subcollection catalog NOTIFICATION_LANGUAGE
```

Options

```
  -c, --catalog string                 Catalog name or id (required)
      --format string                  Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                           Help for notification-languages:get
      --notification-template string   Notification Template name or id (required)
  -o, --org string                     Organization name or id (required)
      --output string                  Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                   scope
  -s, --server string                  management server endpoint (required)
      --space string                   Space name or id (required)
      --subcollection string           subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-languages:list

Notification Languages list operations

Synopsis

Notification Languages list operations

```
apic notification-languages:list [flags]
```

Possible Usages

```
      apic notification-languages:list [flags] --scope space --subcollection space
      apic notification-languages:list [flags] --scope space --subcollection consumer
      apic notification-languages:list [flags] --scope org --subcollection space
      apic notification-languages:list [flags] --scope org --subcollection provider
      apic notification-languages:list [flags] --scope org --subcollection consumer
      apic notification-languages:list [flags] --scope org --subcollection catalog
      apic notification-languages:list [flags] --scope org --subcollection admin
      apic notification-languages:list [flags] --scope cloud --subcollection space
      apic notification-languages:list [flags] --scope cloud --subcollection provider
      apic notification-languages:list [flags] --scope cloud --subcollection consumer
      apic notification-languages:list [flags] --scope cloud --subcollection cloud
      apic notification-languages:list [flags] --scope cloud --subcollection catalog
      apic notification-languages:list [flags] --scope cloud --subcollection admin
      apic notification-languages:list [flags] --scope catalog --subcollection space
      apic notification-languages:list [flags] --scope catalog --subcollection consumer
      apic notification-languages:list [flags] --scope catalog --subcollection catalog
```

Options

```
  -c, --catalog string                 Catalog name or id (required)
      --format string                  Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                           Help for notification-languages:list
      --notification-template string   Notification Template name or id (required)
  -o, --org string                     Organization name or id (required)
      --output string                  Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                   scope
  -s, --server string                  management server endpoint (required)
      --space string                   Space name or id (required)
      --subcollection string           subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-languages:update

Notification Languages update operations

Synopsis

Notification Languages update operations

```
apic notification-languages:update [flags]
```

Possible Usages

```
      apic notification-languages:update [flags] --scope space --subcollection space NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope space --subcollection consumer NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope org --subcollection space NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope org --subcollection provider NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope org --subcollection consumer NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope org --subcollection catalog NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope org --subcollection admin NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection space NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection provider NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection consumer NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection cloud NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection catalog NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope cloud --subcollection admin NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope catalog --subcollection space NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope catalog --subcollection consumer NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
      apic notification-languages:update [flags] --scope catalog --subcollection catalog NOTIFICATION_LANGUAGE
NOTIFICATION_LANGUAGE_FILE
```

Options

```
  -c, --catalog string                 Catalog name or id (required)
      --format string                  Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                           Help for notification-languages:update
      --notification-template string   Notification Template name or id (required)
  -o, --org string                     Organization name or id (required)
      --output string                  Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                   scope
  -s, --server string                  management server endpoint (required)
      --space string                   Space name or id (required)
      --subcollection string           subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-styles

Notification Styles operations

Synopsis

Notification Styles operations

```
apic notification-styles [flags]
```

Options

```
  -h, --help   Help for notification-styles
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-styles:get

Notification Styles get operations

Synopsis

Notification Styles get operations

```
apic notification-styles:get [flags]
```

Possible Usages

```
     apic notification-styles:get [flags] --scope org
     apic notification-styles:get [flags] --scope consumer
     apic notification-styles:get [flags] --scope cloud
     apic notification-styles:get [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for notification-styles:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-styles:update

Notification Styles update operations

Synopsis

Notification Styles update operations

```
apic notification-styles:update [flags]
```

Possible Usages

```
     apic notification-styles:update [flags] --scope org NOTIFICATION_STYLE_FILE
     apic notification-styles:update [flags] --scope consumer NOTIFICATION_STYLE_FILE
     apic notification-styles:update [flags] --scope cloud NOTIFICATION_STYLE_FILE
     apic notification-styles:update [flags] --scope catalog NOTIFICATION_STYLE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for notification-styles:update
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
```

```
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-templates

Notification Templates operations

Synopsis

Notification Templates operations

```
apic notification-templates [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for notification-templates
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-templates:get

Notification Templates get operations

Synopsis

Notification Templates get operations

```
apic notification-templates:get [flags]
```

Possible Usages

```
      apic notification-templates:get [flags] --scope space --subcollection space NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope space --subcollection consumer NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope org --subcollection space NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope org --subcollection provider NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope org --subcollection consumer NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope org --subcollection catalog NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope org --subcollection admin NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection space NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection provider NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection consumer NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection cloud NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection catalog NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope cloud --subcollection admin NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope catalog --subcollection space NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope catalog --subcollection consumer NOTIFICATION_TEMPLATE
      apic notification-templates:get [flags] --scope catalog --subcollection catalog NOTIFICATION_TEMPLATE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
```

```
  -h, --help                    Help for notification-templates:get
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic notification-templates:list

Notification Templates list operations

Synopsis

Notification Templates list operations

```
apic notification-templates:list [flags]
```

Possible Usages

```
      apic notification-templates:list [flags] --scope space --subcollection space
      apic notification-templates:list [flags] --scope space --subcollection consumer
      apic notification-templates:list [flags] --scope org --subcollection space
      apic notification-templates:list [flags] --scope org --subcollection provider
      apic notification-templates:list [flags] --scope org --subcollection consumer
      apic notification-templates:list [flags] --scope org --subcollection catalog
      apic notification-templates:list [flags] --scope org --subcollection admin
      apic notification-templates:list [flags] --scope cloud --subcollection space
      apic notification-templates:list [flags] --scope cloud --subcollection provider
      apic notification-templates:list [flags] --scope cloud --subcollection consumer
      apic notification-templates:list [flags] --scope cloud --subcollection cloud
      apic notification-templates:list [flags] --scope cloud --subcollection catalog
      apic notification-templates:list [flags] --scope cloud --subcollection admin
      apic notification-templates:list [flags] --scope catalog --subcollection space
      apic notification-templates:list [flags] --scope catalog --subcollection consumer
      apic notification-templates:list [flags] --scope catalog --subcollection catalog
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for notification-templates:list
      --limit int32             Maximum number of items to return
      --offset int32            Offset item number from list to begin return
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic notification-templates:list-all

Notification Templates list-all operations

Synopsis

Notification Templates list-all operations

```
apic notification-templates:list-all [flags]
```

Possible Usages

```
      apic notification-templates:list-all [flags] --scope space
      apic notification-templates:list-all [flags] --scope org
      apic notification-templates:list-all [flags] --scope cloud
      apic notification-templates:list-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for notification-templates:list-all
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic notification-templates:update

Notification Templates update operations

Synopsis

Notification Templates update operations

```
apic notification-templates:update [flags]
```

Possible Usages

```
      apic notification-templates:update [flags] --scope space --subcollection space NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope space --subcollection consumer NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope org --subcollection space NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope org --subcollection provider NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope org --subcollection consumer NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope org --subcollection catalog NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope org --subcollection admin NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection space NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection provider NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection consumer NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection cloud NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection catalog NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope cloud --subcollection admin NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope catalog --subcollection space NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope catalog --subcollection consumer NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
      apic notification-templates:update [flags] --scope catalog --subcollection catalog NOTIFICATION_TEMPLATE
NOTIFICATION_TEMPLATE_FILE
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for notification-templates:update
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
```

```
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers

Oauth Providers operations

Synopsis

Oauth Providers operations

```
apic oauth-providers [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for oauth-providers
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:clear

Clear the Oauth Provider objects

Synopsis

Clear the Oauth Provider objects

```
apic oauth-providers:clear [flags]
```

Possible Usages

```
    apic oauth-providers:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:create

Create a Oauth Provider object

Synopsis

Create a Oauth Provider object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic oauth-providers:create [flags]
```

Possible Usages

```
      apic oauth-providers:create [flags] OAUTH_PROVIDER_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:delete

Delete the Oauth Provider object by name or id

Synopsis

Delete the Oauth Provider object by name or id

```
apic oauth-providers:delete [flags]
```

Possible Usages

```
      apic oauth-providers:delete [flags] OAUTH_PROVIDER
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:get

Get the Oauth Provider object by name or id

Synopsis

Get the Oauth Provider object by name or id

```
apic oauth-providers:get [flags]
```

Possible Usages

```
apic oauth-providers:get [flags] OAUTH_PROVIDER
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:list

List the Oauth Provider objects

Synopsis

List the Oauth Provider objects

```
apic oauth-providers:list [flags]
```

Possible Usages

```
apic oauth-providers:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic oauth-providers:update

Update the Oauth Provider object by name or id

Synopsis

Update the Oauth Provider object by name or id

```
Fields allowed but ignored:
- id
- type
- api_version
- scope
- created_at
- updated_at
- url
- org_url
```

```
apic oauth-providers:update [flags]
```

Possible Usages

```
apic oauth-providers:update [flags] OAUTH_PROVIDER OAUTH_PROVIDER_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic org-settings

Org Settings operations

Synopsis

Org Settings operations

```
apic org-settings [flags]
```

Options

```
  -h, --help   Help for org-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic org-settings:get

Get the Organization Setting object

Synopsis

Get the Organization Setting object

```
apic org-settings:get [flags]
```

Possible Usages

```
    apic org-settings:get [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic org-settings:update

Update the Organization Setting object

Synopsis

Update the Organization Setting object

```
      Fields allowed but ignored:
      - name
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url

apic org-settings:update [flags]
```

Possible Usages

```
      apic org-settings:update [flags] ORG_SETTING_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs

Orgs operations

Synopsis

Orgs operations

```
apic orgs [flags]
```

Options

```
      --expand string     List of transient field to expand
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help              Help for orgs
      --limit int32       Maximum number of items to return
      --my                my
      --offset int32      Offset item number from list to begin return
      --org_type string   Type of orgs to return
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string     management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:clear

Clear the Organization objects

Synopsis

Clear the Organization objects

```
apic orgs:clear [flags]
```

Possible Usages

```
      apic orgs:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:create

Create an Organization object

Synopsis

Create an Organization object

```
      Required fields:
      - owner_url

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic orgs:create [flags]
```

Possible Usages

```
      apic orgs:create [flags] ORG_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:delete

Delete the Organization object by name or id

Synopsis

Delete the Organization object by name or id

```
apic orgs:delete [flags]
```

Possible Usages

```
      apic orgs:delete [flags] ORG
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:get

Get the Organization object by name or id

Synopsis

Get the Organization object by name or id

```
apic orgs:get [flags]
```

Possible Usages

```
      apic orgs:get [flags] ORG
```

Options

```
      --expand string   List of transient field to expand
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:list

List the Organization objects

Synopsis

List the Organization objects

```
apic orgs:list [flags]
```

Possible Usages

```
      apic orgs:list [flags] --my
      apic orgs:list [flags]
```

Options

```
      --expand string     List of transient field to expand
      --fields string     List of field names to return
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32       Maximum number of items to return
      --offset int32      Offset item number from list to begin return
      --org_type string   Type of orgs to return
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string     management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic orgs:transfer-owner

Transfer owner to an associate

Synopsis

Transfer owner to an associate

```
apic orgs:transfer-owner [flags]
```

Possible Usages

```
      apic orgs:transfer-owner [flags] ORG TRANSFER_OWNER_FILE
```

Options

```
      --cascade             Cascade the behavior
      --delete_old_owner    Delete old owner
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string       management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic orgs:update

Update the Organization object by name or id

Synopsis

Update the Organization object by name or id

```
      Fields not allowed:
      - name
      - org_type

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic orgs:update [flags]
```

Possible Usages

```
      apic orgs:update [flags] ORG ORG_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic payment-methods

Payment Methods operations

Synopsis

Payment Methods operations

```
apic payment-methods [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for payment-methods
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic payment-methods:create

Create a Payment Method object

Synopsis

Create a Payment Method object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - consumer_org_url
```

```
apic payment-methods:create [flags]
```

Possible Usages

```
      apic payment-methods:create [flags] --space-initiated PAYMENT_METHOD_FILE
      apic payment-methods:create [flags] PAYMENT_METHOD_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic payment-methods:delete

Delete the Payment Method object by name or id

Synopsis

Delete the Payment Method object by name or id

```
apic payment-methods:delete [flags]
```

Possible Usages

```
      apic payment-methods:delete [flags] --space-initiated PAYMENT_METHOD
      apic payment-methods:delete [flags] PAYMENT_METHOD
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic payment-methods:get

Get the Payment Method object by name or id

Synopsis

Get the Payment Method object by name or id

```
apic payment-methods:get [flags]
```

Possible Usages

```
      apic payment-methods:get [flags] --space-initiated PAYMENT_METHOD
      apic payment-methods:get [flags] PAYMENT_METHOD
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic payment-methods:list

List the Payment Method objects

Synopsis

List the Payment Method objects

```
apic payment-methods:list [flags]
```

Possible Usages

```
      apic payment-methods:list [flags] --space-initiated
      apic payment-methods:list [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic payment-methods:update

Update the Payment Method object by name or id

Synopsis

Update the Payment Method object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - consumer_org_url
```

```
apic payment-methods:update [flags]
```

Possible Usages

```
      apic payment-methods:update [flags] --space-initiated PAYMENT_METHOD PAYMENT_METHOD_FILE
      apic payment-methods:update [flags] PAYMENT_METHOD PAYMENT_METHOD_FILE
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic permissions

Permissions operations

Synopsis

Permissions operations

```
apic permissions [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for permissions
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic permissions:get

Permissions get operations

Synopsis

Permissions get operations

```
apic permissions:get [flags]
```

Possible Usages

```
      apic permissions:get [flags] --subcollection provider PERMISSION
      apic permissions:get [flags] --subcollection org PERMISSION
      apic permissions:get [flags] --subcollection deployment PERMISSION
      apic permissions:get [flags] --subcollection consumer PERMISSION
      apic permissions:get [flags] --subcollection cloud PERMISSION
```

Options

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for permissions:get
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic permissions:list

Permissions list operations

Synopsis

Permissions list operations

```
apic permissions:list [flags]
```

Possible Usages

```
      apic permissions:list [flags] --subcollection provider
      apic permissions:list [flags] --subcollection org
      apic permissions:list [flags] --subcollection deployment
      apic permissions:list [flags] --subcollection consumer
      apic permissions:list [flags] --subcollection cloud
      apic permissions:list [flags] --my
```

Options

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for permissions:list
      --limit int32            Maximum number of items to return
      --my                     my
      --offset int32           Offset item number from list to begin return
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic permissions:list-all

List all Permission objects in all collections

Synopsis

List all Permission objects in all collections

```
apic permissions:list-all [flags]
```

Possible Usages

```
      apic permissions:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies

Policies operations

Synopsis

Policies operations

```
apic policies [flags]
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
```

```
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for policies
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:clear

Policies clear operations

Synopsis

Policies clear operations

```
apic policies:clear [flags]
```

Possible Usages

```
      apic policies:clear [flags] --scope space POLICY_NAME
      apic policies:clear [flags] --scope catalog POLICY_NAME
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --confirm string                      Confirmation for critical updates (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                Help for policies:clear
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:clear-all

Policies clear-all operations

Synopsis

Policies clear-all operations

```
apic policies:clear-all [flags]
```

Possible Usages

```
      apic policies:clear-all [flags] --scope space
      apic policies:clear-all [flags] --scope catalog
```

Options

```
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --confirm string                      Confirmation for critical updates (required)
```

```
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for policies:clear-all
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:clone

Policies clone operations

Synopsis

Policies clone operations

```
apic policies:clone [flags]
```

Possible Usages

```
      apic policies:clone [flags] --scope space
      apic policies:clone [flags] --scope org
      apic policies:clone [flags] --scope catalog
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for policies:clone
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:create

Policies create operations

Synopsis

Policies create operations

```
apic policies:create [flags]
```

Possible Usages

```
      apic policies:create [flags] --scope space POLICIES_FILE
      apic policies:create [flags] --scope catalog POLICIES_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for policies:create
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --policy-file string                    Policy document override
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:delete

Policies delete operations

Synopsis

Policies delete operations

```
apic policies:delete [flags]
```

Possible Usages

```
      apic policies:delete [flags] --id --scope space ID
      apic policies:delete [flags] --id --scope catalog ID
      apic policies:delete [flags] --scope space NAME:VERSION
      apic policies:delete [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for policies:delete
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:document

Policies document operations

Synopsis

Policies document operations

```
apic policies:document [flags]
```

Possible Usages

```
      apic policies:document [flags] --id --scope space ID
      apic policies:document [flags] --id --scope org ID
      apic policies:document [flags] --id --scope catalog ID
      apic policies:document [flags] --scope space NAME:VERSION
      apic policies:document [flags] --scope org NAME:VERSION
      apic policies:document [flags] --scope catalog NAME:VERSION
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for policies:document
      --id                                  id
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:get

Policies get operations

Synopsis

Policies get operations

```
apic policies:get [flags]
```

Possible Usages

```
      apic policies:get [flags] --id --scope space ID
      apic policies:get [flags] --id --scope org ID
      apic policies:get [flags] --id --scope catalog ID
      apic policies:get [flags] --scope space NAME:VERSION
      apic policies:get [flags] --scope org NAME:VERSION
      apic policies:get [flags] --scope catalog NAME:VERSION
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return (default "add(policy,implementation)")
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for policies:get
      --id                                  id
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:implementation

Policies implementation operations

Synopsis

Policies implementation operations

```
apic policies:implementation [flags]
```

Possible Usages

```
      apic policies:implementation [flags] --id --scope space ID
      apic policies:implementation [flags] --id --scope catalog ID
      apic policies:implementation [flags] --scope space NAME:VERSION
      apic policies:implementation [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for policies:implementation
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string                          scope
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:list

Policies list operations

Synopsis

Policies list operations

```
apic policies:list [flags]
```

Possible Usages

```
      apic policies:list [flags] --scope space POLICY_NAME
      apic policies:list [flags] --scope org POLICY_NAME
      apic policies:list [flags] --scope catalog POLICY_NAME
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for policies:list
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --space string                        Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:list-all

Policies list-all operations

Synopsis

Policies list-all operations

```
apic policies:list-all [flags]
```

Possible Usages

```
      apic policies:list-all [flags] --scope space
      apic policies:list-all [flags] --scope org
      apic policies:list-all [flags] --scope catalog
```

Options

```
      --availability-zone string             Availability Zone name or id (required)
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string               Gateway Service name or id (required)
  -h, --help                                 Help for policies:list-all
      --limit int32                          Maximum number of items to return
      --offset int32                         Offset item number from list to begin return
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic policies:update

Policies update operations

Synopsis

Policies update operations

```
apic policies:update [flags]
```

Possible Usages

```
      apic policies:update [flags] --id --scope space ID POLICY_FILE [ POLICY-YAML ]
      apic policies:update [flags] --id --scope catalog ID POLICY_FILE [ POLICY-YAML ]
      apic policies:update [flags] --scope space NAME:VERSION POLICY_FILE [ POLICY-YAML ]
      apic policies:update [flags] --scope catalog NAME:VERSION POLICY_FILE [ POLICY-YAML ]
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for policies:update
      --id                                   id
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
      --scope string                         scope
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic portal-services

Portal Services operations

Synopsis

Portal Services operations

```
apic portal-services [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                       Help for portal-services
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
      --scope string               scope
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic portal-services:clear

Clear the Portal Service objects

Synopsis

Clear the Portal Service objects

```
apic portal-services:clear [flags]
```

Possible Usages

```
      apic portal-services:clear [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --confirm string             Confirmation for critical updates (required)
      --force                      Force the operation
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic portal-services:create

Create a Portal Service object

Synopsis

Create a Portal Service object

```
      Required fields:
      - endpoint
      - web_endpoint_base

      Fields not allowed:
```

```
      - owned

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic portal-services:create [flags]
```

Possible Usages

```
      apic portal-services:create [flags] PORTAL_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic portal-services:delete

Delete the Portal Service object by name or id

Synopsis

Delete the Portal Service object by name or id

```
apic portal-services:delete [flags]
```

Possible Usages

```
      apic portal-services:delete [flags] PORTAL_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --force                      Force the operation
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic portal-services:get

Get the Portal Service object by name or id

Synopsis

Get the Portal Service object by name or id

```
apic portal-services:get [flags]
```

Possible Usages

```
      apic portal-services:get [flags] --scope org PORTAL_SERVICE
      apic portal-services:get [flags] PORTAL_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic portal-services:list

List the Portal Service objects

Synopsis

List the Portal Service objects

```
apic portal-services:list [flags]
```

Possible Usages

```
      apic portal-services:list [flags] --scope org
      apic portal-services:list [flags]
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --fields string              List of field names to return
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
      --limit int32                Maximum number of items to return
      --offset int32               Offset item number from list to begin return
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic portal-services:update-credentials

Update the Portal Service configuration

Synopsis

Update the Portal Service configuration

```
apic portal-services:update-credentials [flags]
```

Possible Usages

```
      apic portal-services:update-credentials [flags] PORTAL_SERVICE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic portal-services:update

Update the Portal Service object by name or id

Synopsis

Update the Portal Service object by name or id

```
      Fields not allowed to be null:
      - endpoint_tls_client_profile_url
      - web_endpoint_base

      Fields not allowed:
      - owned
      - endpoint

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - availability_zone_url
```

```
apic portal-services:update [flags]
```

Possible Usages

```
      apic portal-services:update [flags] PORTAL_SERVICE PORTAL_SERVICE_FILE
```

Options

```
      --availability-zone string   Availability Zone name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic primary-events

Primary Events operations

Synopsis

Primary Events operations

```
apic primary-events [flags]
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for primary-events
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --portal-service string               Portal Service name or id (required)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --state string                        State for a webhook event in subscriber queue
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic primary-events:get

Primary Events get operations

Synopsis

Primary Events get operations

```
apic primary-events:get [flags]
```

Possible Usages

```
      apic primary-events:get [flags] --scope portal-service PRIMARY_EVENT
      apic primary-events:get [flags] --scope gateway-service PRIMARY_EVENT
      apic primary-events:get [flags] --scope configured-gateway-service PRIMARY_EVENT
      apic primary-events:get [flags] --scope cloud PRIMARY_EVENT
      apic primary-events:get [flags] --scope catalog PRIMARY_EVENT
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for primary-events:get
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, use - for STDOUT. (default: cwd)
      --portal-service string               Portal Service name or id (required)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic primary-events:list

Primary Events list operations

Synopsis

Primary Events list operations

```
apic primary-events:list [flags]
```

Possible Usages

```
      apic primary-events:list [flags] --scope portal-service
      apic primary-events:list [flags] --scope gateway-service
      apic primary-events:list [flags] --scope configured-gateway-service
      apic primary-events:list [flags] --scope cloud
      apic primary-events:list [flags] --scope catalog
```

Options

```
      --availability-zone string            Availability Zone name or id (required)
  -c, --catalog string                      Catalog name or id (required)
      --configured-gateway-service string   Configured Gateway Service name or id (required)
      --fields string                       List of field names to return
      --format string                       Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
      --gateway-service string              Gateway Service name or id (required)
  -h, --help                                Help for primary-events:list
      --limit int32                         Maximum number of items to return
      --offset int32                        Offset item number from list to begin return
  -o, --org string                          Organization name or id (required)
      --output string                       Write file(s) to directory, instead of STDOUT (default "-")
      --portal-service string               Portal Service name or id (required)
      --scope string                        scope
  -s, --server string                       management server endpoint (required)
      --state string                        State for a webhook event in subscriber queue
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products

Products operations

Synopsis

Products operations

```
apic products [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --expand string    List of transient field to expand
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:clear

Products clear operations

Synopsis

Products clear operations

```
apic products:clear [flags]
```

Possible Usages

```
    apic products:clear [flags] --scope space PRODUCT_NAME
    apic products:clear [flags] --scope catalog PRODUCT_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:clear-all

Products clear-all operations

Synopsis

Products clear-all operations

```
apic products:clear-all [flags]
```

Possible Usages

```
    apic products:clear-all [flags] --scope space
    apic products:clear-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:clear-all
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:clone

Products clone operations

Synopsis

Products clone operations

```
apic products:clone [flags]
```

Possible Usages

```
    apic products:clone [flags] --scope space
    apic products:clone [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --expand string    List of transient field to expand
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:clone
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:delete

Products delete operations

Synopsis

Products delete operations

```
apic products:delete [flags]
```

Possible Usages

```
    apic products:delete [flags] --id --scope space ID
    apic products:delete [flags] --id --scope catalog ID
    apic products:delete [flags] --scope space NAME:VERSION
    apic products:delete [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:delete
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:document

Products document operations

Synopsis

Products document operations

```
apic products:document [flags]
```

Possible Usages

```
    apic products:document [flags] --id --scope space ID
    apic products:document [flags] --id --scope catalog ID
    apic products:document [flags] --scope space NAME:VERSION
    apic products:document [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:document
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:execute-migration-target

Products execute-migration-target operations

Synopsis

Products execute-migration-target operations

```
apic products:execute-migration-target [flags]
```

Possible Usages

```
    apic products:execute-migration-target [flags] --id --scope space ID
    apic products:execute-migration-target [flags] --id --scope catalog ID
    apic products:execute-migration-target [flags] --scope space NAME:VERSION
    apic products:execute-migration-target [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:execute-migration-target
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:get

Products get operations

Synopsis

Products get operations

```
apic products:get [flags]
```

Possible Usages

```
    apic products:get [flags] --id --scope space ID
    apic products:get [flags] --id --scope catalog ID
    apic products:get [flags] --scope space NAME:VERSION
    apic products:get [flags] --scope catalog NAME:VERSION
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return (default "add(catalog_product)")
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:get
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:list

Products list operations

Synopsis

Products list operations

```
apic products:list [flags]
```

Possible Usages

```
    apic products:list [flags] --scope space PRODUCT_NAME
    apic products:list [flags] --scope catalog PRODUCT_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --expand string    List of transient field to expand
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:list-all

Products list-all operations

Synopsis

Products list-all operations

```
apic products:list-all [flags]
```

Possible Usages

```
    apic products:list-all [flags] --scope space
    apic products:list-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --expand string    List of transient field to expand
```

```
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:list-all
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:migrate-subscriptions

Products migrate-subscriptions operations

Synopsis

Products migrate-subscriptions operations

```
apic products:migrate-subscriptions [flags]
```

Possible Usages

```
    apic products:migrate-subscriptions [flags] --id --scope space ID MIGRATE_SUBSCRIPTION_SUBSET_FILE
    apic products:migrate-subscriptions [flags] --id --scope catalog ID MIGRATE_SUBSCRIPTION_SUBSET_FILE
    apic products:migrate-subscriptions [flags] --scope space NAME:VERSION MIGRATE_SUBSCRIPTION_SUBSET_FILE
    apic products:migrate-subscriptions [flags] --scope catalog NAME:VERSION MIGRATE_SUBSCRIPTION_SUBSET_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:migrate-subscriptions
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic products:publish

Publish the product

Synopsis

Publish the product

```
apic products:publish [flags]
```

Possible Usages

```
    apic products:publish [flags] --stage --scope space PRODUCTS_FILE
    apic products:publish [flags] --stage PRODUCTS_FILE
    apic products:publish [flags] --scope space PRODUCTS_FILE
    apic products:publish [flags] PRODUCTS_FILE
```

Options

```
  -c, --catalog string             Catalog name or id (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
      --gateway_services string    The list of gateway service names to support partial publishing
      --migrate_subscriptions      Migrate subscription when republish product
  -o, --org string                 Organization name or id (required)
      --output string              Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic products:replace

Products replace operations

Synopsis

Products replace operations

Create a mapping file

if you are replacing a Product, the mapping file specifies the Product that you want to replace, and the mapping of the Plans from the source Product to the target Product.

The PRODUCT_PLAN_MAPPING_FILE takes the following form:

```
product_url: https:/server/api/catalogs/{id}/products/{id}
plans:
- source: {source_plan_name_1}
  target: {target_plan_name_1}
- source: {source_plan_name_2}
  target: {target_plan_name_2}
                          .
                          .
                          .
```

`Hot-replace` version 1.0.0 with 1.0.1

```
apic products:replace climbon:1.0.1 PRODUCT_PLAN_MAPPING_FILE
```

The Product specified on the command line is the replacement Product. For example, if you are replacing climbon:1.0.0 with climbon:1.0.1 then the Product specified on the command line is climbon:1.0.1, and the product_url property inside the plan mapping file specifies the URL of climbon:1.0.0.

```
apic products:replace [flags]
```

Possible Usages

```
      apic products:replace [flags] --id --scope space ID PRODUCT_PLAN_MAPPING_FILE
      apic products:replace [flags] --id --scope catalog ID PRODUCT_PLAN_MAPPING_FILE
      apic products:replace [flags] --scope space NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
      apic products:replace [flags] --scope catalog NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:replace
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic products:set-migration-target

Products set-migration-target operations

Synopsis

Products set-migration-target operations

```
apic products:set-migration-target [flags]
```

Possible Usages

```
apic products:set-migration-target [flags] --id --scope space ID PRODUCT_PLAN_MAPPING_FILE
apic products:set-migration-target [flags] --id --scope catalog ID PRODUCT_PLAN_MAPPING_FILE
apic products:set-migration-target [flags] --scope space NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
apic products:set-migration-target [flags] --scope catalog NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
```

Options

```
-c, --catalog string   Catalog name or id (required)
    --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
-h, --help             Help for products:set-migration-target
    --id               id
-o, --org string       Organization name or id (required)
    --output string    Write file(s) to directory, instead of STDOUT (default "-")
    --scope string     scope
-s, --server string    management server endpoint (required)
    --space string     Space name or id (required)
```

Options inherited from parent commands

```
    --accept-license        Accept the license for API Connect
    --debug                 Enable debug output
    --debug-output string   Write debug output to file
    --live-help             Enable or disable tracking of limited usage information
-m, --mode string           Toolkit operation mode (default "apim")
```

apic products:supersede

Products supersede operations

Synopsis

Products supersede operations

```
apic products:supersede [flags]
```

Possible Usages

```
apic products:supersede [flags] --id --scope space ID PRODUCT_PLAN_MAPPING_FILE
apic products:supersede [flags] --id --scope catalog ID PRODUCT_PLAN_MAPPING_FILE
apic products:supersede [flags] --scope space NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
apic products:supersede [flags] --scope catalog NAME:VERSION PRODUCT_PLAN_MAPPING_FILE
```

Options

```
-c, --catalog string   Catalog name or id (required)
    --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
-h, --help             Help for products:supersede
    --id               id
-o, --org string       Organization name or id (required)
    --output string    Write file(s) to directory, instead of STDOUT (default "-")
    --scope string     scope
-s, --server string    management server endpoint (required)
    --space string     Space name or id (required)
```

Options inherited from parent commands

```
    --accept-license        Accept the license for API Connect
    --debug                 Enable debug output
    --debug-output string   Write debug output to file
    --live-help             Enable or disable tracking of limited usage information
-m, --mode string           Toolkit operation mode (default "apim")
```

apic products:update

Products update operations

Synopsis

Products update operations

```
apic products:update [flags]
```

Possible Usages

```
apic products:update [flags] --id --scope space ID PRODUCT_FILE
apic products:update [flags] --id --scope catalog ID PRODUCT_FILE
apic products:update [flags] --scope space NAME:VERSION PRODUCT_FILE
apic products:update [flags] --scope catalog NAME:VERSION PRODUCT_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for products:update
      --id               id
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties

Properties operations

Synopsis

Properties operations

```
apic properties [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:clear

Properties clear operations

Synopsis

Properties clear operations

```
apic properties:clear [flags]
```

Possible Usages

```
      apic properties:clear [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:create

Properties create operations

Synopsis

Properties create operations

```
apic properties:create [flags]
```

Possible Usages

```
      apic properties:create [flags] --scope catalog NAME_VALUE_PAIRS_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties:create
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:delete

Properties delete operations

Synopsis

Properties delete operations

```
apic properties:delete [flags]
```

Possible Usages

```
      apic properties:delete [flags] --scope catalog PROPERTY_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties:delete
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
```

```
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:get

Properties get operations

Synopsis

Properties get operations

```
apic properties:get [flags]
```

Possible Usages

```
      apic properties:get [flags] --scope catalog PROPERTY_NAME
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:list

Properties list operations

Synopsis

Properties list operations

```
apic properties:list [flags]
```

Possible Usages

```
      apic properties:list [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for properties:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic properties:update

Properties update operations

Synopsis

Properties update operations

```
apic properties:update [flags]
```

Possible Usages

```
      apic properties:update [flags] --scope catalog PROPERTY_NAME PROPERTY_FILE
```

Options

```
  -c, --catalog string    Catalog name or id (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for properties:update
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
      --scope string      scope
  -s, --server string     management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations

Registrations operations

Synopsis

Registrations operations

```
apic registrations [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for registrations
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:clear

Clear the Registration objects

Synopsis

Clear the Registration objects

```
apic registrations:clear [flags]
```

Possible Usages

```
apic registrations:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:create

Create a Registration object

Synopsis

Create a Registration object

```
Fields allowed but ignored:
- id
- type
- api_version
- scope
- created_at
- updated_at
- url
```

```
apic registrations:create [flags]
```

Possible Usages

```
apic registrations:create [flags] REGISTRATION_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:delete

Delete the Registration object by name or id

Synopsis

Delete the Registration object by name or id

```
apic registrations:delete [flags]
```

Possible Usages

```
apic registrations:delete [flags] REGISTRATION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:get

Get the Registration object by name or id

Synopsis

Get the Registration object by name or id

```
apic registrations:get [flags]
```

Possible Usages

```
      apic registrations:get [flags] REGISTRATION
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:list

List the Registration objects

Synopsis

List the Registration objects

```
apic registrations:list [flags]
```

Possible Usages

```
      apic registrations:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic registrations:update

Update the Registration object by name or id

Synopsis

Update the Registration object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic registrations:update [flags]
```

Possible Usages

```
      apic registrations:update [flags] REGISTRATION REGISTRATION_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults

Role Defaults operations

Synopsis

Role Defaults operations

```
apic role-defaults [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for role-defaults
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults:clear

Role Defaults clear operations

Synopsis

Role Defaults clear operations

```
apic role-defaults:clear [flags]
```

Possible Usages

```
      apic role-defaults:clear [flags] --scope cloud --subcollection provider
```

Options

```
      --confirm string          Confirmation for critical updates (required)
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for role-defaults:clear
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults:create

Role Defaults create operations

Synopsis

Role Defaults create operations

```
apic role-defaults:create [flags]
```

Possible Usages

```
      apic role-defaults:create [flags] --scope cloud --subcollection provider ROLE_DEFAULT_FILE
```

Options

```
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for role-defaults:create
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults:delete

Role Defaults delete operations

Synopsis

Role Defaults delete operations

```
apic role-defaults:delete [flags]
```

Possible Usages

```
      apic role-defaults:delete [flags] --scope cloud --subcollection provider ROLE_DEFAULT
```

Options

```
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for role-defaults:delete
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic role-defaults:get

Role Defaults get operations

Synopsis

Role Defaults get operations

```
apic role-defaults:get [flags]
```

Possible Usages

```
      apic role-defaults:get [flags] --scope space --subcollection consumer ROLE_DEFAULT
      apic role-defaults:get [flags] --scope cloud --subcollection provider ROLE_DEFAULT
      apic role-defaults:get [flags] --scope cloud --subcollection consumer ROLE_DEFAULT
      apic role-defaults:get [flags] --scope catalog --subcollection consumer ROLE_DEFAULT
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for role-defaults:get
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic role-defaults:list

Role Defaults list operations

Synopsis

Role Defaults list operations

```
apic role-defaults:list [flags]
```

Possible Usages

```
      apic role-defaults:list [flags] --scope space --subcollection consumer
      apic role-defaults:list [flags] --scope cloud --subcollection provider
      apic role-defaults:list [flags] --scope cloud --subcollection consumer
      apic role-defaults:list [flags] --scope catalog --subcollection consumer
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for role-defaults:list
      --limit int32             Maximum number of items to return
      --offset int32            Offset item number from list to begin return
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --subcollection string    subcollection
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults:list-all

Role Defaults list-all operations

Synopsis

Role Defaults list-all operations

```
apic role-defaults:list-all [flags]
```

Possible Usages

```
    apic role-defaults:list-all [flags] --scope space
    apic role-defaults:list-all [flags] --scope cloud
    apic role-defaults:list-all [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for role-defaults:list-all
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic role-defaults:update

Role Defaults update operations

Synopsis

Role Defaults update operations

```
apic role-defaults:update [flags]
```

Possible Usages

```
    apic role-defaults:update [flags] --scope cloud --subcollection provider ROLE_DEFAULT ROLE_DEFAULT_FILE
```

Options

```
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for role-defaults:update
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --subcollection string   subcollection
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic roles

Roles operations

Synopsis

Roles operations

```
apic roles [flags]
```

Options

```
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for roles
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
      --scope string           scope
  -s, --server string          management server endpoint (required)
      --space string           Space name or id (required)
      --space-initiated        space-initiated
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic roles:clear

Roles clear operations

Synopsis

Roles clear operations

```
apic roles:clear [flags]
```

Possible Usages

```
      apic roles:clear [flags] --scope org
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for roles:clear
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic roles:create

Roles create operations

Synopsis

Roles create operations

```
apic roles:create [flags]
```

Possible Usages

```
      apic roles:create [flags] --scope org ROLE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for roles:create
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string    scope
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic roles:delete

Roles delete operations

Synopsis

Roles delete operations

```
apic roles:delete [flags]
```

Possible Usages

```
      apic roles:delete [flags] --scope org ROLE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for roles:delete
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string    scope
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic roles:get

Roles get operations

Synopsis

Roles get operations

```
apic roles:get [flags]
```

Possible Usages

```
      apic roles:get [flags] --scope consumer-org --space-initiated ROLE
      apic roles:get [flags] --scope space ROLE
      apic roles:get [flags] --scope org ROLE
      apic roles:get [flags] --scope consumer-org ROLE
      apic roles:get [flags] --scope catalog ROLE
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --consumer-org string     Consumer Organization name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for roles:get
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --space-initiated         space-initiated
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic roles:list

Roles list operations

Synopsis

Roles list operations

```
apic roles:list [flags]
```

Possible Usages

```
      apic roles:list [flags] --scope consumer-org --space-initiated
      apic roles:list [flags] --scope space
      apic roles:list [flags] --scope org
      apic roles:list [flags] --scope consumer-org
      apic roles:list [flags] --scope catalog
```

Options

```
  -c, --catalog string          Catalog name or id (required)
      --consumer-org string     Consumer Organization name or id (required)
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                    Help for roles:list
      --limit int32             Maximum number of items to return
      --offset int32            Offset item number from list to begin return
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
      --scope string            scope
  -s, --server string           management server endpoint (required)
      --space string            Space name or id (required)
      --space-initiated         space-initiated
```

Options inherited from parent commands

```
      --accept-license          Accept the license for API Connect
      --debug                   Enable debug output
      --debug-output string     Write debug output to file
      --live-help               Enable or disable tracking of limited usage information
  -m, --mode string             Toolkit operation mode (default "apim")
```

apic roles:update

Roles update operations

Synopsis

Roles update operations

```
apic roles:update [flags]
```

Possible Usages

```
apic roles:update [flags] --scope space ROLE ROLE_FILE
apic roles:update [flags] --scope org ROLE ROLE_FILE
apic roles:update [flags] --scope catalog ROLE ROLE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for roles:update
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services

Services operations

Synopsis

Services operations

```
apic services [flags]
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for services
      --limit int32                          Maximum number of items to return
      --offset int32                         Offset item number from list to begin return
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
      --space-initiated                      space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:clear

Services clear operations

Synopsis

Services clear operations

```
apic services:clear [flags]
```

Possible Usages

```
      apic services:clear [flags] --space-initiated SERVICE_NAME
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --confirm string                       Confirmation for critical updates (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for services:clear
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
      --space-initiated                      space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:clear-all

Services clear-all operations

Synopsis

Services clear-all operations

```
apic services:clear-all [flags]
```

Possible Usages

```
      apic services:clear-all [flags] --space-initiated
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --confirm string                       Confirmation for critical updates (required)
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for services:clear-all
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
      --space-initiated                      space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:create

Services create operations

Synopsis

Services create operations

```
apic services:create [flags]
```

Possible Usages

```
apic services:create [flags] --space-initiated SERVICE_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for services:create
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
      --space-initiated                       space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:delete

Services delete operations

Synopsis

Services delete operations

```
apic services:delete [flags]
```

Possible Usages

```
apic services:delete [flags] --id --space-initiated ID
apic services:delete [flags] --space-initiated NAME:VERSION
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for services:delete
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
      --space-initiated                       space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:get

Services get operations

Synopsis

Services get operations

```
apic services:get [flags]
```

Possible Usages

```
      apic services:get [flags] --id --space-initiated ID
      apic services:get [flags] --space-initiated NAME:VERSION
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for services:get
      --id                                   id
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
      --space-initiated                      space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:list

Services list operations

Synopsis

Services list operations

```
apic services:list [flags]
```

Possible Usages

```
      apic services:list [flags] --space-initiated SERVICE_NAME
```

Options

```
  -c, --catalog string                       Catalog name or id (required)
      --configured-gateway-service string    Configured Gateway Service name or id (required)
      --fields string                        List of field names to return
      --format string                        Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                 Help for services:list
      --limit int32                          Maximum number of items to return
      --offset int32                         Offset item number from list to begin return
  -o, --org string                           Organization name or id (required)
      --output string                        Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                        management server endpoint (required)
      --space string                         Space name or id (required)
      --space-initiated                      space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:list-all

Services list-all operations

Synopsis

Services list-all operations

```
apic services:list-all [flags]
```

Possible Usages

```
      apic services:list-all [flags] --space-initiated
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --fields string                         List of field names to return
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for services:list-all
      --limit int32                           Maximum number of items to return
      --offset int32                          Offset item number from list to begin return
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
      --space-initiated                       space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic services:update

Services update operations

Synopsis

Services update operations

```
apic services:update [flags]
```

Possible Usages

```
      apic services:update [flags] --id --space-initiated ID SERVICE_FILE
      apic services:update [flags] --space-initiated NAME:VERSION SERVICE_FILE
```

Options

```
  -c, --catalog string                        Catalog name or id (required)
      --configured-gateway-service string     Configured Gateway Service name or id (required)
      --format string                         Output format. One of [json yaml octet-stream go-template=... go-template-
file=...], defaults to yaml.
  -h, --help                                  Help for services:update
      --id                                    id
  -o, --org string                            Organization name or id (required)
      --output string                         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string                         management server endpoint (required)
      --space string                          Space name or id (required)
      --space-initiated                       space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic space-settings

Space Settings operations

Synopsis

Space Settings operations

```
apic space-settings [flags]
```

Options

```
  -h, --help   Help for space-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic space-settings:get

Get the Space Setting object

Synopsis

Get the Space Setting object

```
apic space-settings:get [flags]
```

Possible Usages

```
      apic space-settings:get [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic space-settings:update

Update the Space Setting object

Synopsis

Update the Space Setting object

```
      Fields allowed but ignored:
      - name
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - space_url
```

```
apic space-settings:update [flags]
```

Possible Usages

```
      apic space-settings:update [flags] SPACE_SETTING_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic spaces

Spaces operations

Synopsis

Spaces operations

```
apic spaces [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for spaces
      --limit int32      Maximum number of items to return
      --my               my
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic spaces:clear

Clear the Space objects

Synopsis

Clear the Space objects

```
apic spaces:clear [flags]
```

Possible Usages

```
      apic spaces:clear [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic spaces:create

Create a Space object

Synopsis

Create a Space object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
```

```
apic spaces:create [flags]
```

Possible Usages

```
      apic spaces:create [flags] SPACE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:delete

Delete the Space object by name or id

Synopsis

Delete the Space object by name or id

```
apic spaces:delete [flags]
```

Possible Usages

```
      apic spaces:delete [flags] SPACE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:email-to-owners

Send email to owners of consumer organizations

Synopsis

Send email to owners of consumer organizations, given consumer org and consumer group urls. For consumer group, email owners of all consumer orgs in the group.

```
apic spaces:email-to-owners [flags]
```

Possible Usages

```
apic spaces:email-to-owners [flags] SPACE EMAIL_TO_OWNERS_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:get

Get the Space object by name or id

Synopsis

Get the Space object by name or id

```
apic spaces:get [flags]
```

Possible Usages

```
apic spaces:get [flags] SPACE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:list

List the Space objects

Synopsis

List the Space objects

```
apic spaces:list [flags]
```

Possible Usages

```
apic spaces:list [flags] --my
apic spaces:list [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:transfer-owner

Transfer owner to an associate

Synopsis

Transfer owner to an associate

```
apic spaces:transfer-owner [flags]
```

Possible Usages

```
    apic spaces:transfer-owner [flags] SPACE TRANSFER_OWNER_FILE
```

Options

```
      --cascade             Cascade the behavior
  -c, --catalog string      Catalog name or id (required)
      --delete_old_owner    Delete old owner
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string       management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic spaces:update

Update the Space object by name or id

Synopsis

Update the Space object by name or id

```
    Fields allowed but ignored:
    - id
    - type
    - api_version
    - scope
    - created_at
    - updated_at
    - url
    - org_url
    - catalog_url
```

```
apic spaces:update [flags]
```

Possible Usages

```
    apic spaces:update [flags] SPACE SPACE_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriber-events

Subscriber Events operations

Synopsis

Subscriber Events operations

```
apic subscriber-events [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for subscriber-events
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --state string     State for a webhook event in subscriber queue
      --webhook string   Webhook name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriber-events:get

Subscriber Events get operations

Synopsis

Subscriber Events get operations

```
apic subscriber-events:get [flags]
```

Possible Usages

```
    apic subscriber-events:get [flags] --scope cloud SUBSCRIBER_EVENT
    apic subscriber-events:get [flags] --scope catalog SUBSCRIBER_EVENT
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for subscriber-events:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --webhook string   Webhook name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriber-events:list

Subscriber Events list operations

Synopsis

Subscriber Events list operations

```
apic subscriber-events:list [flags]
```

Possible Usages

```
      apic subscriber-events:list [flags] --scope cloud
      apic subscriber-events:list [flags] --scope catalog
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for subscriber-events:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --state string     State for a webhook event in subscriber queue
      --webhook string   Webhook name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriptions

Subscriptions operations

Synopsis

Subscriptions operations

```
apic subscriptions [flags]
```

Options

```
  -a, --app string                Application name or id (required)
  -c, --catalog string            Catalog name or id (required)
      --consumer-org string       Consumer Organization name or id (required)
      --consumer_org_url string   Consumer Org Url
      --expand string             List of transient field to expand
      --fields string             List of field names to return
      --format string             Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                      Help for subscriptions
      --limit int32               Maximum number of items to return
      --offset int32              Offset item number from list to begin return
  -o, --org string                Organization name or id (required)
      --output string             Write file(s) to directory, instead of STDOUT (default "-")
      --plan string               Plan Name
      --product_url string        Product Url
      --scope string              scope
  -s, --server string             management server endpoint (required)
      --space string              Space name or id (required)
      --space-initiated           space-initiated
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriptions:clear

Clear the Subscription objects

Synopsis

Clear the Subscription objects

```
apic subscriptions:clear [flags]
```

Possible Usages

```
      apic subscriptions:clear [flags] --space-initiated
      apic subscriptions:clear [flags]
```

Options

```
  -a, --app string            Application name or id (required)
  -c, --catalog string        Catalog name or id (required)
      --confirm string        Confirmation for critical updates (required)
      --consumer-org string   Consumer Organization name or id (required)
      --format string         Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string            Organization name or id (required)
      --output string         Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriptions:create

Create a Subscription object

Synopsis

Create a Subscription object

```
      Required fields:
      - product_url
      - plan

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
      - app_url
```

```
apic subscriptions:create [flags]
```

Possible Usages

```
      apic subscriptions:create [flags] --space-initiated SUBSCRIPTION_FILE
      apic subscriptions:create [flags] SUBSCRIPTION_FILE
```

Options

```
  -a, --app string            Application name or id (required)
  -c, --catalog string        Catalog name or id (required)
      --consumer-org string   Consumer Organization name or id (required)
```

```
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic subscriptions:delete

Delete the Subscription object by name or id

Synopsis

Delete the Subscription object by name or id

```
apic subscriptions:delete [flags]
```

Possible Usages

```
      apic subscriptions:delete [flags] --space-initiated SUBSCRIPTION
      apic subscriptions:delete [flags] SUBSCRIPTION
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic subscriptions:get

Get the Subscription object by name or id

Synopsis

Get the Subscription object by name or id

```
apic subscriptions:get [flags]
```

Possible Usages

```
      apic subscriptions:get [flags] --space-initiated SUBSCRIPTION
      apic subscriptions:get [flags] SUBSCRIPTION
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriptions:list

List the Subscription objects

Synopsis

List the Subscription objects

```
apic subscriptions:list [flags]
```

Possible Usages

```
      apic subscriptions:list [flags] --space-initiated
      apic subscriptions:list [flags] --scope space
      apic subscriptions:list [flags] --scope catalog
      apic subscriptions:list [flags]
```

Options

```
  -a, --app string            Application name or id (required)
  -c, --catalog string        Catalog name or id (required)
      --consumer-org string   Consumer Organization name or id (required)
      --fields string         List of field names to return
      --format string         Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32           Maximum number of items to return
      --offset int32          Offset item number from list to begin return
  -o, --org string            Organization name or id (required)
      --output string         Write file(s) to directory, instead of STDOUT (default "-")
      --plan string           Plan Name
      --product_url string    Product Url
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic subscriptions:update

Update the Subscription object by name or id

Synopsis

Update the Subscription object by name or id

```
      Fields not allowed:
      - billing_identifiers
      - plan_title

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - catalog_url
      - consumer_org_url
      - app_url
```

```
apic subscriptions:update [flags]
```

Possible Usages

```
      apic subscriptions:update [flags] --space-initiated SUBSCRIPTION SUBSCRIPTION_FILE
      apic subscriptions:update [flags] SUBSCRIPTION SUBSCRIPTION_FILE
```

Options

```
  -a, --app string             Application name or id (required)
  -c, --catalog string         Catalog name or id (required)
      --consumer-org string    Consumer Organization name or id (required)
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic task-queues

Task Queues operations

Synopsis

Task Queues operations

```
apic task-queues [flags]
```

Options

```
      --apply_filter     Filter tasks
      --cascade          Cascade the behavior
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for task-queues
      --kind string      kind of item
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --query string     Add query to request
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --state string     State for a webhook event in subscriber queue
      --target string    target for the task
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic task-queues:get

Task Queues get operations

Synopsis

Task Queues get operations

```
apic task-queues:get [flags]
```

Possible Usages

```
      apic task-queues:get [flags] --scope cloud TASK_QUEUE
```

Options

```
      --cascade          Cascade the behavior
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
```

```
  -h, --help             Help for task-queues:get
      --kind string      kind of item
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --query string     Add query to request
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --state string     State for a webhook event in subscriber queue
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic task-queues:list

Task Queues list operations

Synopsis

Task Queues list operations

```
apic task-queues:list [flags]
```

Possible Usages

```
      apic task-queues:list [flags] --scope cloud
```

Options

```
      --apply_filter     Filter tasks
      --cascade          Cascade the behavior
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for task-queues:list
      --kind string      kind of item
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --query string     Add query to request
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --state string     State for a webhook event in subscriber queue
      --target string    target for the task
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic tasks

Tasks operations

Synopsis

Tasks operations

```
apic tasks [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for tasks
      --limit int32      Maximum number of items to return
      --my               my
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --originated       originated
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
```

```
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tasks:get

Tasks get operations

Synopsis

Tasks get operations

```
apic tasks:get [flags]
```

Possible Usages

```
      apic tasks:get [flags] --scope space TASK
      apic tasks:get [flags] --scope catalog TASK
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for tasks:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tasks:list

Tasks list operations

Synopsis

Tasks list operations

```
apic tasks:list [flags]
```

Possible Usages

```
      apic tasks:list [flags] --scope space
      apic tasks:list [flags] --scope catalog
      apic tasks:list [flags] --originated
      apic tasks:list [flags] --my
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for tasks:list
      --limit int32      Maximum number of items to return
      --my               my
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --originated       originated
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
```

```
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tasks:update

Tasks update operations

Synopsis

Tasks update operations

```
apic tasks:update [flags]
```

Possible Usages

```
      apic tasks:update [flags] --scope space TASK TASK_FILE
      apic tasks:update [flags] --scope catalog TASK TASK_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for tasks:update
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
      --space string     Space name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic test

Runs a test and returns

Synopsis

apic-test is a CLI for testing your APIs

```
apic test [flags]
```

Options

```
  -h, --help            Help for test
      --output string   ATM test
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles

Tls Client Profiles operations

Synopsis

Tls Client Profiles operations

```
apic tls-client-profiles [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for tls-client-profiles
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles:clear

Clear the TLS Client Profile objects

Synopsis

Clear the TLS Client Profile objects

```
apic tls-client-profiles:clear [flags]
```

Possible Usages

```
    apic tls-client-profiles:clear [flags] TLS_CLIENT_PROFILE_NAME
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles:clear-all

Clear all TLS Client Profile objects in all collections

Synopsis

Clear all TLS Client Profile objects in all collections

```
apic tls-client-profiles:clear-all [flags]
```

Possible Usages

```
    apic tls-client-profiles:clear-all [flags]
```

Options

```
      --cascade          Cascade the behavior
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
```

```
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles:create

Create a TLS Client Profile object

Synopsis

Create a TLS Client Profile object

```
      Fields not allowed:
      - owned

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic tls-client-profiles:create [flags]
```

Possible Usages

```
      apic tls-client-profiles:create [flags] TLS_CLIENT_PROFILE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles:delete

Delete a TLS Client Profile

Synopsis

Delete a TLS Client Profile

```
apic tls-client-profiles:delete [flags]
```

Possible Usages

```
      apic tls-client-profiles:delete [flags] --id ID
      apic tls-client-profiles:delete [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic tls-client-profiles:get

Get the TLS Client Profile object by name and version

Synopsis

Get the TLS Client Profile object by name and version

```
apic tls-client-profiles:get [flags]
```

Possible Usages

```
      apic tls-client-profiles:get [flags] --id ID
      apic tls-client-profiles:get [flags] NAME:VERSION
```

Options

```
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic tls-client-profiles:list

List the TLS Client Profile objects

Synopsis

List the TLS Client Profile objects

```
apic tls-client-profiles:list [flags]
```

Possible Usages

```
      apic tls-client-profiles:list [flags] TLS_CLIENT_PROFILE_NAME
```

Options

```
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic tls-client-profiles:list-all

List all TLS Client Profile objects in all collections

Synopsis

List all TLS Client Profile objects in all collections

```
apic tls-client-profiles:list-all [flags]
```

Possible Usages

```
apic tls-client-profiles:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-client-profiles:update

Update the TLS Client Profile object by name and version

Synopsis

Update the TLS Client Profile object by name and version

```
Fields not allowed:
- owned

Fields not allowed to be null:
- ciphers
- protocols

Fields allowed but ignored:
- id
- type
- api_version
- scope
- created_at
- updated_at
- url
- org_url
```

```
apic tls-client-profiles:update [flags]
```

Possible Usages

```
apic tls-client-profiles:update [flags] --id ID TLS_CLIENT_PROFILE_FILE
apic tls-client-profiles:update [flags] NAME:VERSION TLS_CLIENT_PROFILE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles

Tls Server Profiles operations

Synopsis

Tls Server Profiles operations

```
apic tls-server-profiles [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for tls-server-profiles
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:clear

Clear the TLS Server Profile objects

Synopsis

Clear the TLS Server Profile objects

```
apic tls-server-profiles:clear [flags]
```

Possible Usages

```
      apic tls-server-profiles:clear [flags] TLS_SERVER_PROFILE_NAME
```

Options

```
      --cascade           Cascade the behavior
      --confirm string    Confirmation for critical updates (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string     management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:clear-all

Clear all TLS Server Profile objects in all collections

Synopsis

Clear all TLS Server Profile objects in all collections

```
apic tls-server-profiles:clear-all [flags]
```

Possible Usages

```
      apic tls-server-profiles:clear-all [flags]
```

Options

```
      --cascade           Cascade the behavior
      --confirm string    Confirmation for critical updates (required)
      --format string     Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string        Organization name or id (required)
      --output string     Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string     management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:create

Create a TLS Server Profile object

Synopsis

Create a TLS Server Profile object

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic tls-server-profiles:create [flags]
```

Possible Usages

```
      apic tls-server-profiles:create [flags] TLS_SERVER_PROFILE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:delete

Delete a TLS Server Profile

Synopsis

Delete a TLS Server Profile

```
apic tls-server-profiles:delete [flags]
```

Possible Usages

```
      apic tls-server-profiles:delete [flags] --id ID
      apic tls-server-profiles:delete [flags] NAME:VERSION
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:get

Get the TLS Server Profile object by name and version

Synopsis

Get the TLS Server Profile object by name and version

```
apic tls-server-profiles:get [flags]
```

Possible Usages

```
      apic tls-server-profiles:get [flags] --id ID
      apic tls-server-profiles:get [flags] NAME:VERSION
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:list

List the TLS Server Profile objects

Synopsis

List the TLS Server Profile objects

```
apic tls-server-profiles:list [flags]
```

Possible Usages

```
      apic tls-server-profiles:list [flags] TLS_SERVER_PROFILE_NAME
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:list-all

List all TLS Server Profile objects in all collections

Synopsis

List all TLS Server Profile objects in all collections

```
apic tls-server-profiles:list-all [flags]
```

Possible Usages

```
      apic tls-server-profiles:list-all [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic tls-server-profiles:update

Update the TLS Server Profile object by name and version

Synopsis

Update the TLS Server Profile object by name and version

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic tls-server-profiles:update [flags]
```

Possible Usages

```
      apic tls-server-profiles:update [flags] --id ID TLS_SERVER_PROFILE_FILE
      apic tls-server-profiles:update [flags] NAME:VERSION TLS_SERVER_PROFILE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores

Truststores operations

Synopsis

Truststores operations

```
apic truststores [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for truststores
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:clear

Clear the Truststore objects

Synopsis

Clear the Truststore objects

```
apic truststores:clear [flags]
```

Possible Usages

```
apic truststores:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:create

Create a Truststore object

Synopsis

Create a Truststore object

```
Fields allowed but ignored:
- id
- type
- api_version
- scope
- created_at
- updated_at
- url
```

```
apic truststores:create [flags]
```

Possible Usages

```
apic truststores:create [flags] TRUSTSTORE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:delete

Delete the Truststore object by name or id

Synopsis

Delete the Truststore object by name or id

```
apic truststores:delete [flags]
```

Possible Usages

```
apic truststores:delete [flags] TRUSTSTORE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:get

Get the Truststore object by name or id

Synopsis

Get the Truststore object by name or id

```
apic truststores:get [flags]
```

Possible Usages

```
apic truststores:get [flags] TRUSTSTORE
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:list

List the Truststore objects

Synopsis

List the Truststore objects

```
apic truststores:list [flags]
```

Possible Usages

```
      apic truststores:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic truststores:update

Update the Truststore object by name or id

Synopsis

Update the Truststore object by name or id

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic truststores:update [flags]
```

Possible Usages

```
      apic truststores:update [flags] TRUSTSTORE TRUSTSTORE_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries

User Registries operations

Synopsis

User Registries operations

```
apic user-registries [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for user-registries
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:clear

Clear the User Registry objects

Synopsis

Clear the User Registry objects

```
apic user-registries:clear [flags]
```

Possible Usages

```
apic user-registries:clear [flags]
```

Options

```
      --confirm string   Confirmation for critical updates (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:create

Create a User Registry object

Synopsis

Create a User Registry object

```
Required fields:
- integration_url

Fields not allowed:
- owned
```

```
      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic user-registries:create [flags]
```

Possible Usages

```
      apic user-registries:create [flags] USER_REGISTRY_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:delete

Delete the User Registry object by name or id

Synopsis

Delete the User Registry object by name or id

```
apic user-registries:delete [flags]
```

Possible Usages

```
      apic user-registries:delete [flags] USER_REGISTRY
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:execute

Execute a User Registry operation

Synopsis

Execute a User Registry operation

```
apic user-registries:execute [flags]
```

Possible Usages

```
      apic user-registries:execute [flags] --test-config-only USER_REGISTRY_EXECUTE_FILE
      apic user-registries:execute [flags] USER_REGISTRY
```

Options

```
      --format string       Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --operation string    Operation to perform on the user registry (get_base_dn_list, validate_password) (required)
  -o, --org string          Organization name or id (required)
      --output string       Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string       management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:get

Get the User Registry object by name or id

Synopsis

Get the User Registry object by name or id

```
apic user-registries:get [flags]
```

Possible Usages

```
      apic user-registries:get [flags] USER_REGISTRY
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:list

List the User Registry objects

Synopsis

List the User Registry objects

```
apic user-registries:list [flags]
```

Possible Usages

```
      apic user-registries:list [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:search

Search for users in the user registry

Synopsis

Search for users in the user registry

```
apic user-registries:search [flags]
```

Possible Usages

```
      apic user-registries:search [flags] USER_REGISTRY USER_SEARCH_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:test-connection

Test a User Registry connection

Synopsis

Test a User Registry connection

```
apic user-registries:test-connection [flags]
```

Possible Usages

```
      apic user-registries:test-connection [flags] --test-config-only USER_REGISTRY_TEST_CONNECTION_FILE
      apic user-registries:test-connection [flags] USER_REGISTRY TEST_CONNECTION_CREDENTIALS_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registries:update

Update the User Registry object by name or id

Synopsis

Update the User Registry object by name or id

```
      Fields not allowed:
      - owned
      - registry_type
      - user_registry_managed
      - correlation_data
      - integration_url

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
```

```
apic user-registries:update [flags]
```

Possible Usages

```
      apic user-registries:update [flags] USER_REGISTRY USER_REGISTRY_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registry-settings

User Registry Settings operations

Synopsis

User Registry Settings operations

```
apic user-registry-settings [flags]
```

Options

```
  -h, --help   Help for user-registry-settings
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registry-settings:get

Get the User Registry Setting object

Synopsis

Get the User Registry Setting object

```
apic user-registry-settings:get [flags]
```

Possible Usages

```
      apic user-registry-settings:get [flags]
```

Options

```
      --fields string   List of field names to return
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic user-registry-settings:update

Update the User Registry Setting object

Synopsis

Update the User Registry Setting object

```
      Fields allowed but ignored:
      - name
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
```

```
apic user-registry-settings:update [flags]
```

Possible Usages

```
      apic user-registry-settings:update [flags] USER_REGISTRY_SETTING_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users

Users operations

Synopsis

Users operations

```
apic users [flags]
```

Options

```
      --fields string          List of field names to return
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                   Help for users
      --limit int32            Maximum number of items to return
      --offset int32           Offset item number from list to begin return
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --user-registry string   User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users:clear

Clear the User objects

Synopsis

Clear the User objects

```
apic users:clear [flags]
```

Possible Usages

```
      apic users:clear [flags]
```

Options

```
      --confirm string          Confirmation for critical updates (required)
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --user-registry string    User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users:create

Create a User object

Synopsis

Create a User object

```
      Required fields:
      - username

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - user_registry_url
```

```
apic users:create [flags]
```

Possible Usages

```
      apic users:create [flags] USER_FILE
```

Options

```
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --user-registry string    User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic users:delete

Delete the User object by name or id

Synopsis

Delete the User object by name or id

```
apic users:delete [flags]
```

Possible Usages

```
      apic users:delete [flags] USER
```

Options

```
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --user-registry string    User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic users:get

Get the User object by name or id

Synopsis

Get the User object by name or id

```
apic users:get [flags]
```

Possible Usages

```
      apic users:get [flags] USER
```

Options

```
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, use - for STDOUT. (default: cwd)
  -s, --server string           management server endpoint (required)
      --user-registry string    User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic users:list

List the User objects

Synopsis

List the User objects

```
apic users:list [flags]
```

Possible Usages

```
apic users:list [flags]
```

Options

```
      --fields string           List of field names to return
      --format string           Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
      --limit int32             Maximum number of items to return
      --offset int32            Offset item number from list to begin return
  -o, --org string              Organization name or id (required)
      --output string           Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string           management server endpoint (required)
      --user-registry string    User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users:request-password-reset

Send reset password link

Synopsis

Send reset password link

```
apic users:request-password-reset [flags]
```

Possible Usages

```
apic users:request-password-reset [flags] REQUEST_PASSWORD_RESET_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users:search-provider

Users search-provider operations

Synopsis

Users search-provider operations

```
apic users:search-provider [flags]
```

Possible Usages

```
apic users:search-provider [flags] --scope org USER_SEARCH_FILE
```

Options

```
      --format string   Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for users:search-provider
      --limit int32     Maximum number of items to return
      --offset int32    Offset item number from list to begin return
  -o, --org string      Organization name or id (required)
      --output string   Write file(s) to directory, instead of STDOUT (default "-")
      --scope string    scope
  -s, --server string   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic users:update

Update the User object by name or id

Synopsis

Update the User object by name or id

```
      Fields not allowed:
      - identity_provider
      - username
      - salt
      - correlation_data
      - force_password_change
      - last_login_at

      Fields allowed but ignored:
      - id
      - type
      - api_version
      - scope
      - created_at
      - updated_at
      - url
      - org_url
      - user_registry_url
```

```
apic users:update [flags]
```

Possible Usages

```
      apic users:update [flags] USER USER_FILE
```

Options

```
      --format string          Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -o, --org string             Organization name or id (required)
      --output string          Write file(s) to directory, instead of STDOUT (default "-")
  -s, --server string          management server endpoint (required)
      --user-registry string   User Registry name or id (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic validate

Validate an API or product definition

Synopsis

Validate an API or product definition

```
apic validate [FILE] [flags]
```

Examples

```
Validate an API definition
  $ apic validate routes.yaml
  Validated routes.yaml API definition [routes:1.0]
Validate an API definition without IBM extensions
  $ apic validate --no-extensions routes.yaml
  Validated routes.yaml API definition [routes:1.0]
Validate a product definition and its referenced APIs
  $ apic validate climb-on.yaml
  Validated climb-on.yaml product definition [climb-on:1.0.0]
  Validated routes.yaml API definition [valid:1.0]
Validate a product definition without validating its referenced APIs
  $ apic validate --product-only climb-on.yaml
  Validated climb-on.yaml product definition [climb-on:1.0.0]
```

Options

```
  -h, --help             Help for validate
      --no-extensions    for API definitions, do not validate against IBM Swagger extensions
  -p, --product-only     for products definitions, do not validate referenced APIs
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic version

Get the APIConnect toolkit version

Synopsis

Get the APIConnect toolkit version

```
apic version [flags]
```

Options

```
  -h, --help             Help for version
  -s, --server string    management server endpoint
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic webhooks

Webhooks operations

Synopsis

Webhooks operations

```
apic webhooks [flags]
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for webhooks
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic webhooks:get

Webhooks get operations

Synopsis

Webhooks get operations

```
apic webhooks:get [flags]
```

Possible Usages

```
      apic webhooks:get [flags] --scope cloud WEBHOOK
      apic webhooks:get [flags] --scope catalog WEBHOOK
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for webhooks:get
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, use - for STDOUT. (default: cwd)
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic webhooks:list

Webhooks list operations

Synopsis

Webhooks list operations

```
apic webhooks:list [flags]
```

Possible Usages

```
      apic webhooks:list [flags] --scope cloud
      apic webhooks:list [flags] --scope catalog
      apic webhooks:list [flags] --scope all
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --fields string    List of field names to return
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for webhooks:list
      --limit int32      Maximum number of items to return
      --offset int32     Offset item number from list to begin return
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic webhooks:update

Webhooks update operations

Synopsis

Webhooks update operations

```
apic webhooks:update [flags]
```

Possible Usages

```
      apic webhooks:update [flags] --scope cloud WEBHOOK WEBHOOK_FILE
      apic webhooks:update [flags] --scope catalog WEBHOOK WEBHOOK_FILE
```

Options

```
  -c, --catalog string   Catalog name or id (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for webhooks:update
  -o, --org string       Organization name or id (required)
      --output string    Write file(s) to directory, instead of STDOUT (default "-")
      --scope string     scope
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic wsdl

Introspect WSDL files

Synopsis

Introspect a WSDL file displaying all its services $ apic wsdl WSDL_FILE $ apic wsdl:introspect WSDL_FILE

```
apic wsdl [flags]
```

Options

```
  -h, --help   Help for wsdl
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic wsdl:introspect

Introspect a WSDL file displaying all its services

Synopsis

Introspect a WSDL file displaying all its services

```
apic wsdl:introspect WSDL_FILE [flags]
```

Options

```
  -h, --help   Help for wsdl:introspect
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic

APIConnect toolkit 61bd16316ae32e02b6cd8e72d4b18e02560de16f (Built 2020-09-23T09:51:50Z)

Synopsis

APIConnect toolkit 61bd16316ae32e02b6cd8e72d4b18e02560de16f (Built 2020-09-23T09:51:50Z)

```
apic [flags]
```

Options

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
  -h, --help                  Help for apic
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api

Api operations

Synopsis

Api operations

```
apic api --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for api
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:add-attachment

Add an attachment for the given api.

Synopsis

Add an attachment for a api within the developer portal of the provided org and catalog. A api can hold 10 attachments. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:add-attachment --mode portaladmin [flags]
```

Options

```
      --attachment_description string   A description of the attachment to be displayed to users.
      --attachment_name string          The name given to the attachment once it has been uploaded e.g. my-file.txt (required)
  -c, --catalog string                  Name or ID of the catalog that the site belongs to. (required)
      --format string                   Output format. One of [json yaml go-template=... go-template-file=...], defaults to
```

```
yaml.
  -h, --help                            Help for api:add-attachment
  -o, --org string                      Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string                   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:add-tag

Add a tag for the given api.

Synopsis

Add a tag for a api within the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:add-tag --mode portaladmin [flags]
```

Options

```
  -c, --catalog string    Name or ID of the catalog that the site belongs to. (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for api:add-tag
  -o, --org string        Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string     management server endpoint (required)
      --tag_name string   The tag name e.g. top_level_element/next_level_element (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:get

Get a specific api from a developer portal.

Synopsis

Get a specific api from the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for api:get
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:get-document

Get a specific entire api document from a portal.

Synopsis

Get a specific entire api document from the portal of the provided organization and catalog. The id or name:version of a specific api needs to be provided. For example, 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:get-document --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for api:get-document
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:list

Shows a list of apis from a site.

Synopsis

Show a list of apis of the provided org and catalog.

```
apic api:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for api:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic api:set-icon

Set an icon image for the given api.

Synopsis

Set an icon image for a api within the developer portal of the provided org and catalog. The existing icon will be overwritten. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic api:set-icon --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --format string             Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                      Help for api:set-icon
      --icon_description string   A description of the icon to be displayed to users. Used as alt text for the image.
(required)
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apic-config

Apic Config operations

Synopsis

Apic Config operations

```
apic apic-config --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for apic-config
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic apic-config:get

Get the configuration object that the developer portal holds.

Synopsis

Get the configuration object that the developer portal holds that was sent by APIM.

```
apic apic-config:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for apic-config:get
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic application

Application operations

Synopsis

Application operations

```
apic application --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for application
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic application:get

Get a specific application from a developer portal.

Synopsis

Get a specific application from the developer portal of the provided org and catalog. The id of a specific consumer organization needs to be provided.

```
apic application:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for application:get
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic application:list

Shows a list of applications from a site.

Synopsis

Show a list of applications of the provided org and catalog.

```
apic application:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for application:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic backups

Backups operations

Synopsis

Backups operations

```
apic backups --mode portaladmin [flags]
```

Options

```
      --format string               Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                        Help for backups
```

```
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic backups:list

List the backups present on the remote backup server.

Synopsis

Lists all the backups that are currently present on the remote backup server, if one has been set.

```
apic backups:list --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for backups:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config

Manage configuration variables

Synopsis

Manage configuration variables

```
apic config --mode portaladmin [flags]
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:clear

Delete all configuration variables

Synopsis

Delete all configuration variables

```
apic config:clear --mode portaladmin [flags]
```

Examples

```
$ apic config:clear --mode consumer
Deleted config catalog
Deleted config org
Deleted config space

$ apic config:clear --global --mode consumer
Deleted config catalog
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:clear
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:delete

Delete a configuration variable

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/consumer-api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:delete NAME... --mode portaladmin [flags]
```

Examples

```
$ apic config:delete catalog --mode consumer
Deleted config catalog

$ apic config:delete org --mode consumer
Deleted config org
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:delete
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:get

Get a configuration variable

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/consumer-api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:get NAME --mode portaladmin [flags]
```

Examples

```
$ apic config:get catalog --mode consumer
catalog: https://mgmnthost.com/consumer-api/catalogs/climbon/sb

$ apic config:get org --mode consumer
org: https://mgmnthost.com/consumer-api/orgs/climbon
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:get
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:list

List the application and global configuration variables

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/consumer-api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:list --mode portaladmin [flags]
```

Examples

```
$ apic config --mode consumer
catalog: https://mgmnthost.com/consumer-api/catalogs/climbon/sb
org: https://mgmnthost.com/consumer-api/orgs/climbon

$ apic config --global --mode consumer
catalog: https://mgmnthost.com/consumer-api/catalogs/climbon/sb
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:list
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic config:set

Set or update configuration variables

Synopsis

Configuration variables:

catalog The catalog configuration variable should be set to the URI of an API Connect catalog. The value of the catalog provides the default identity of a catalog for all the commands that are used to manage aspects of a catalog. The default values defined by the catalog's URI can be overridden using the --server, --organization, and --catalog command line options. The catalog URI has the form: https://MANAGEMENT_SERVER/consumer-api/catalogs/CONSUMER_ORG_NAME/CATALOG_NAME

org The default value of org defined by the app's or catalog's URI can be set using the org URI. The org URI has the form: https://MANAGEMENT_SERVER/consumer-api/orgs/CONSUMER_ORG_NAME

space The space configuration variable should be set to the URI of an API Connect space. The value of the space provides the default identity of a space for all the commands that are used to manage aspects of a space. The default values defined by the spaces's URI can be overridden using the --server, --organization, --catalog, and --space command line options. The space URI has the form: https://MANAGEMENT_SERVER/consumer-api/spaces/CONSUMER_ORG_NAME/CATALOG_NAME/SPACE_NAME

cloud The cloud configuration variable should be set to the management server URI. The value of the cloud variable provides default server URI for cloud admin commands. It can be overridden using the --server and --mode command line options. The cloud URI has the form: https://MANAGEMENT_SERVER/consumer-api/

mode The value of mode configruation variable defines the toolkit operation mode. It can be overridden using the --mode command line option. The value can be set to apim or consumer

template-path List of places to look for handlebar templates

template-default-api Defines the default api template

template-default-product Defines the default product template

```
apic config:set NAME=VALUE ... --mode portaladmin [flags]
```

Examples

```
$ apic config:set --mode consumer catalog=https://mgmnthost.com/consumer-api/catalogs/climbon/sb
catalog: https://mgmnthost.com/consumer-api/catalogs/climbon/sb

$ apic config:set --mode consumer org=https://mgmnthost2.com/consumer-api/orgs/hikeon
org: https://mgmnthost2.com/consumer-api/orgs/hikeon
```

Options

```
  -g, --global   list the global configuration variables
  -h, --help     Help for config:set
  -l, --local    list the local application configuration variables
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-org

Consumer Org operations

Synopsis

Consumer Org operations

```
apic consumer-org --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for consumer-org
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-org:get

Get a specific consumer org from a developer portal.

Synopsis

Get a specific consumer org from the developer portal of the provided org and catalog. The id of a specific consumer organization needs to be provided.

```
apic consumer-org:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for consumer-org:get
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic consumer-org:list

Shows a list of consumer orgs from a site.

Synopsis

Show a list of consumer orgs of the provided org and catalog.

```
apic consumer-org:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for consumer-org:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content

Content operations

Synopsis

Content operations

```
apic content --mode portaladmin [flags]
```

Options

```
      --bundle string        The bundle type for a given entity type. E.g. "page" for the "node" entity type.
  -c, --catalog string       Name or ID of the catalog that the site belongs to. (required)
      --entity_type string   Type of entity you want to perform this operation on. To find out entity/content types see the
content list-types route. (required)
      --format string        Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                 Help for content
  -o, --org string           Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string        management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:create-export

Create a task that exports a tgz containing the site content.

Synopsis

Creates a task that exports a tgz containing the site content. You can export specific entity types and specific ids or you can generate a tar containing all content. Depending on your site's content please be advised that an export all can take a long time to complete. If called via the toolkit CLI, it will automatically poll the response every 30 seconds up to 25 minutes.

```
apic content:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string       Name or ID of the catalog that the site belongs to. (required)
      --entity_ids string    A comma separated list of entity ids.
      --entity_type string   Type of entity you want to perform this operation on. To find out entity/content types see the
content list-types route. (required)
      --format string        Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                 Help for content:create-export
      --no-poll              Do not poll the created task and just return the task ID
  -o, --org string           Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string        management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:create-import

Create a task that imports an archive of site content.

Synopsis

Creates a task that imports an archive of site content. The status of this task can be tracked with the /content/TASK_ID/status route. Depending on your site's content please be advised that an import can take a long time to complete. If called via the toolkit CLI, it will automatically poll the response every 30 seconds up to 25 minutes.

```
apic content:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:create-import
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:delete-export

If running, cancels a content export task.

Synopsis

Cancels any currently running content export tasks, and deletes any related artifacts that have been generated.

```
apic content:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:delete-import

If running, cancels a content import task.

Synopsis

Cancels any currently running content import tasks, and deletes any related artifacts that have been generated.

```
apic content:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic content:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:get-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:get-export

When ready, streams a content export artifact.

Synopsis

When ready, streams a content export artifact back to the related task id that is provided in the arguments.

```
apic content:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:get-import-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic content:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:get-import-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:list

Lists all of the entities in your site for the given entity type and bundle.

Synopsis

Lists all of the entities in your site for the given entity type and bundle.

```
apic content:list --mode portaladmin [flags]
```

Options

```
      --bundle string        The bundle type for a given entity type. E.g. "page" for the "node" entity type.
  -c, --catalog string       Name or ID of the catalog that the site belongs to. (required)
      --entity_type string   Type of entity you want to perform this operation on. To find out entity/content types see the
content list-types route. (required)
      --format string        Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                 Help for content:list
  -o, --org string           Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string        management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic content:list-types

Lists the exportable content types on your site.

Synopsis

Lists the content types that are exportable from your site.

```
apic content:list-types --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for content:list-types
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license   Accept the license for API Connect
      --debug            Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module

Custom Module operations

Synopsis

Custom Module operations

```
apic custom-module --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for custom-module
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:create-export

Create a task that exports an archive that contains the custom modules of a site.

Synopsis

You can create a task to export an archive of the custom modules of a site. You can then use that archive to quickly and simply override the custom modules on another site.

```
apic custom-module:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:create-export
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:create-import

Create a task that imports an archive that contains the custom modules of a site.

Synopsis

You can create a task to import an archive of the custom modules of a site. You can use the imported archive to quickly and simply override the custom modules for the specified site.

```
apic custom-module:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:create-import
      --no-poll          Do not poll the created task and just return the task ID
```

```
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:delete-export

If running, cancels a custom module export task.

Synopsis

If running, cancels the custom module export task and deletes any related artifacts that have been generated.

```
apic custom-module:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:delete-import

If running, cancels a custom module import task.

Synopsis

If running, cancels the custom module import task and deletes any related artifacts that have been generated.

```
apic custom-module:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:get-export

When ready, streams a custom module export artifact.

Synopsis

When ready, streams a custom module export artifact back to the related task id that is provided in the arguments.

```
apic custom-module:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic custom-module:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:get-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-module:get-import-status

Get the result of the custom module import task.

Synopsis

Get the result of the custom module import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-module:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-module:get-import-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme

Custom Theme operations

Synopsis

Custom Theme operations

```
apic custom-theme --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for custom-theme
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:create-export

Create a task that exports an archive that contains the custom themes of a site.

Synopsis

You can create a task to export an archive of the custom themes of a site. You can then use that archive to quickly and simply override the custom themes on another site.

```
apic custom-theme:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:create-export
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:create-import

Create a task that imports an archive that contains the custom themes of a site.

Synopsis

You can create a task to import an archive of the custom themes of a site. You can use the imported archive to quickly and simply override the custom themes for the specified site.

```
apic custom-theme:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:create-import
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:delete-export

If running, cancels a custom theme export task.

Synopsis

If running, cancels the custom theme export task and deletes any related artifacts that have been generated. This does not delete any custom themes.

```
apic custom-theme:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:delete-import

Cancels a running custom theme import task.

Synopsis

Cancels the custom theme import task and deletes any generated artifacts.

```
apic custom-theme:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:get-export

When ready, streams a custom theme export artifact.

Synopsis

When ready, streams a custom theme export artifact back to the related task id that is provided in the arguments.

```
apic custom-theme:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic custom-theme:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:get-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-theme:get-import-status

Get the result of the custom theme import task.

Synopsis

Get the result of the custom theme import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-theme:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-theme:get-import-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation

Custom Translation operations

Synopsis

Custom Translation operations

```
apic custom-translation --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for custom-translation
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation:create-export

Create a task that exports an archive that contains the custom translation of a site.

Synopsis

You can create a task to export an archive of the custom translation of a site. You can then use that archive to quickly and simply override the custom translations on another site.

```
apic custom-translation:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string     Name or ID of the catalog that the site belongs to. (required)
      --format string      Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help               Help for custom-translation:create-export
      --langcodes string   A comma separated list of language codes e.g. "es,zh-hans"
      --no-poll            Do not poll the created task and just return the task ID
  -o, --org string         Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string      management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation:create-import

Create a task that imports an archive that contains the custom translations of a site.

Synopsis

You can create a task to import an archive of the custom translations of a site. You can use the imported archive to simply override the custom translations for the specified site. This command may take several minutes to complete.

```
apic custom-translation:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-translation:create-import
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation:delete-export

If running, cancels a custom translation export task.

Synopsis

If running, cancels the custom translation export task and deletes any related artifacts that have been generated. This does not delete any custom translations.

```
apic custom-translation:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-translation:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation:delete-import

If running, cancels a custom translation import task.

Synopsis

If running, cancels the custom translation import task and deletes any related artifacts that have been generated. This does not delete any custom translations.

```
apic custom-translation:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-translation:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-translation:get-export

When ready, streams a custom translation export artifact.

Synopsis

When ready, streams a custom translation export artifact back to the related task id that is provided in the arguments.

```
apic custom-translation:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for custom-translation:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic custom-translation:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic custom-translation:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string    Name or ID of the catalog that the site belongs to. (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for custom-translation:get-export-status
  -o, --org string        Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string     management server endpoint (required)
      --task_id string    ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic custom-translation:get-import-status

Get the result of the custom translation import task.

Synopsis

Get the result of the custom translation import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic custom-translation:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string    Name or ID of the catalog that the site belongs to. (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for custom-translation:get-import-status
  -o, --org string        Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string     management server endpoint (required)
      --task_id string    ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic custom-webserver-page

Custom Web Server Page operations

Synopsis

Custom Web Server Page operations

```
apic custom-webserver-page --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for custom-webserver-page
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-webserver-page:delete

Delete the custom HTML for a web server page.

Synopsis

Delete the custom web server HTML content of a specific web server page (index, 404, 40x, 50x). (Will fallback to a default.)

```
apic custom-webserver-page:delete --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for custom-webserver-page:delete
      --page_type string             The page type you want to run this command against. Values: index, 404, 40x, 50x
(required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-webserver-page:get

Get the HTML for a web server page.

Synopsis

Get the web server HTML content of a specific web server page (index, 404, 40x, 50x).

```
apic custom-webserver-page:get --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for custom-webserver-page:get
      --page_type string             The page type you want to run this command against. Values: index, 404, 40x, 50x
(required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic custom-webserver-page:set

Set a custom HTML page for a web server page type.

Synopsis

Set the custom web server HTML content of a specific web server page (index, 404, 40x, 50x). Maximum size of 8 MB.

```
apic custom-webserver-page:set --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for custom-webserver-page:set
      --page_type string             The page type you want to run this command against. Values: index, 404, 40x, 50x
(required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-config

Drupal Config operations

Synopsis

Drupal Config operations

```
apic drupal-config --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for drupal-config
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --prefix string    The config prefix. For example, "system". No prefix will return all names in the system.
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-config:delete

Delete the Config object.

Synopsis

Delete the Config object or a specific config key.

```
apic drupal-config:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string        Name or ID of the catalog that the site belongs to. (required)
      --config_name string    The config object name, for example "system.site". (required)
      --format string         Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                  Help for drupal-config:delete
      --config_key string     A config key, for example "page.front".
  -o, --org string            Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-config:get

Get the Config object

Synopsis

Get the Config object or a specific config key value.

```
apic drupal-config:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string       Name or ID of the catalog that the site belongs to. (required)
      --config_key string    A config key, for example "page.front".
      --config_name string   The config object name, for example "system.site". (required)
      --format string        Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                 Help for drupal-config:get
  -o, --org string           Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string        management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license       Accept the license for API Connect
      --debug                Enable debug output
      --debug-output string  Write debug output to file
      --live-help            Enable or disable tracking of limited usage information
  -m, --mode string          Toolkit operation mode (default "apim")
```

apic drupal-config:list

List the config names by prefix

Synopsis

List the config names by prefix or provide no prefix to see all.

```
apic drupal-config:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for drupal-config:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --prefix string    The config prefix. For example, "system". No prefix will return all names in the system.
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license       Accept the license for API Connect
      --debug                Enable debug output
      --debug-output string  Write debug output to file
      --live-help            Enable or disable tracking of limited usage information
  -m, --mode string          Toolkit operation mode (default "apim")
```

apic drupal-config:set

Set a config value.

Synopsis

Can set a new config value or update an already existing key-value.

```
apic drupal-config:set --mode portaladmin [flags]
```

Options

```
  -c, --catalog string        Name or ID of the catalog that the site belongs to. (required)
      --config_key string     A config key, for example "page.front". (required)
      --config_name string    The config object name, for example "system.site". (required)
      --config_value string   The value to assign to the config key. (required)
      --format string         Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                  Help for drupal-config:set
```

```
  -o, --org string            Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string         management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state

Drupal State operations

Synopsis

Drupal State operations

```
apic drupal-state --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for drupal-state
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:delete

Delete the State key.

Synopsis

Delete the specific state key.

```
apic drupal-state:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string     Name or ID of the catalog that the site belongs to. (required)
      --format string      Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help               Help for drupal-state:delete
  -o, --org string         Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string      management server endpoint (required)
      --state_key string   The state key, for example "system.cron_last". (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:get

Get the State key

Synopsis

Get the State key value.

```
apic drupal-state:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string     Name or ID of the catalog that the site belongs to. (required)
      --format string      Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help               Help for drupal-state:get
  -o, --org string         Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string      management server endpoint (required)
      --state_key string   The state key, for example "system.cron_last". (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:set

Set a state key value.

Synopsis

Can set a new state value or update an already existing key-value.

```
apic drupal-state:set --mode portaladmin [flags]
```

Options

```
  -c, --catalog string         Name or ID of the catalog that the site belongs to. (required)
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for drupal-state:set
      --input_format string    The input format of the value for the state key. Values: string, integer, float, boolean, json,
yaml. [default: auto]
  -o, --org string             Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string          management server endpoint (required)
      --state_key string       The state key, for example "system.cron_last". (required)
      --state_value string     The value to assign to the state key. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state

Drupal State operations

Synopsis

Drupal State operations

```
apic drupal-state --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for drupal-state
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:delete

Delete the State key.

Synopsis

Delete the specific state key.

```
apic drupal-state:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string     Name or ID of the catalog that the site belongs to. (required)
      --format string      Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help               Help for drupal-state:delete
  -o, --org string         Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string      management server endpoint (required)
      --state_key string   The state key, for example "system.cron_last". (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:get

Get the State key

Synopsis

Get the State key value.

```
apic drupal-state:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string     Name or ID of the catalog that the site belongs to. (required)
      --format string      Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help               Help for drupal-state:get
  -o, --org string         Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string      management server endpoint (required)
      --state_key string   The state key, for example "system.cron_last". (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic drupal-state:set

Set a state key value.

Synopsis

Can set a new state value or update an already existing key-value.

```
apic drupal-state:set --mode portaladmin [flags]
```

Options

```
  -c, --catalog string         Name or ID of the catalog that the site belongs to. (required)
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for drupal-state:set
      --input_format string    The input format of the value for the state key. Values: string, integer, float, boolean, json,
yaml. [default: auto]
  -o, --org string             Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string          management server endpoint (required)
      --state_key string       The state key, for example "system.cron_last". (required)
      --state_value string     The value to assign to the state key. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entity

Entity operations

Synopsis

Entity operations

```
apic entity --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for entity
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic entity:count

Shows a count of entities from a site.

Synopsis

Show a count of entities of the provided org and catalog.

```
apic entity:count --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for entity:count
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity

Export Entity operations

Synopsis

Export Entity operations

```
apic export-entity --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for export-entity
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:add-content

Add content to an export entity

Synopsis

Add content to an existing export entity. Provide the content type and IDs that you wish to add to this export entity.

```
apic export-entity:add-content --mode portaladmin [flags]
```

Options

```
  -c, --catalog string             Name or ID of the catalog that the site belongs to. (required)
      --entity_ids string          A comma separated list of entity ids.
      --entity_type string         Type of entity you want to perform this operation on. To find out entity/content types see
the content list-types route. (required)
      --export_entity_id string    ID of the export entity (required)
      --format string              Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                       Help for export-entity:add-content
  -o, --org string                 Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string              management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:create

Create an export entity

Synopsis

Create a new export entity. This export entity can be populated with content which can subsequently be exported.

```
apic export-entity:create --mode portaladmin [flags]
```

Options

```
  -c, --catalog string              Name or ID of the catalog that the site belongs to. (required)
      --export_entity_name string   name of the export entity
      --format string               Output format. One of [json yaml octet-stream go-template=... go-template-file=...],
defaults to yaml.
  -h, --help                        Help for export-entity:create
  -o, --org string                  Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string               management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:delete-launch-export

If running, cancels a export entity launch export task.

Synopsis

If running, cancels a export entity launch export task and deletes any related artifacts that have been generated.

```
apic export-entity:delete-launch-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for export-entity:delete-launch-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:delete

Delete an export entity

Synopsis

Delete an export entity by its ID.

```
apic export-entity:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --export_entity_id string   ID of the export entity (required)
      --format string             Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                      Help for export-entity:delete
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:get-launch-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic export-entity:get-launch-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for export-entity:get-launch-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:get-launch-export

When ready, streams a content export artifact generated from an export entity.

Synopsis

When ready, streams a content export artifact generated from an export entity back to the related task id that is provided in the arguments.

```
apic export-entity:get-launch-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for export-entity:get-launch-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:get

Get an export entity

Synopsis

Get an export entity by its ID, which will list the content that will be exported as part of this export entity.

```
apic export-entity:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --export_entity_id string   ID of the export entity (required)
      --format string             Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                      Help for export-entity:get
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic export-entity:launch

Launch a export entity

Synopsis

Launch an export entity which will create an export tgz of all the entities contained in the export. If called via the toolkit CLI, it will automatically poll the response every 30 seconds up to 25 minutes.

```
apic export-entity:launch --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --export_entity_id string   ID of the export entity (required)
      --format string             Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                      Help for export-entity:launch
      --no-poll                   Do not poll the created task and just return the task ID
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic export-entity:list

List export entities

Synopsis

List export entities within your Developer Portal. Export Entities contain a defined list of content to be exported when launched.

```
apic export-entity:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for export-entity:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic export-entity:remove-content

Remove content from an export entity

Synopsis

Remove content from an export entity. Provide the content type and IDs that you wish to remove from this export entity.

```
apic export-entity:remove-content --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --entity_ids string         A comma separated list of entity ids.
      --entity_type string        Type of entity you want to perform this operation on. To find out entity/content types see
the content list-types route. (required)
      --export_entity_id string   ID of the export entity (required)
      --format string             Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults
to yaml.
  -h, --help                      Help for export-entity:remove-content
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic factory-reset

Factory Reset operations

Synopsis

Factory Reset operations

```
apic factory-reset --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for factory-reset
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic factory-reset:delete

Factory resets the Developer Portal.

Synopsis

Performs a factory reset of the Developer Portal, deleting the portal service and all portal sites. WARNING: This command is irreversible - please ensure you have backups configured.

```
apic factory-reset:delete --mode portaladmin [flags]
```

Options

```
      --execute_reset string              Set to true to trigger the Developer Portal reset
      --format string                     Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                              Help for factory-reset:delete
      --portal_service_endpoint string    The URL endpoint of the portal service (required)
  -s, --server string                     Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic forums

Forums operations

Synopsis

Forums operations

```
apic forums --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for forums
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic forums:disable

Disable the forum module for a given site

Synopsis

Removes all forums vocabulary taxonomy terms before disabling the forum module for a given site.

```
apic forums:disable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for forums:disable
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic forums:enable

Enable the forum module for a given site

Synopsis

Enables the forum module for a given site.

```
apic forums:enable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for forums:enable
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic ip-security-enabled

IP Security Enabled operations.

Synopsis

IP Security Enabled operations. Toggle the IP security on your Developer Portal between enabled and disabled. IP security is enabled by default.

```
apic ip-security-enabled --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for ip-security-enabled
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic ip-security-enabled:update

Enable or disable IP security functionality within the Developer Portal.

Synopsis

Enable or disable IP security functionality within the Developer Portal. IP security is enabled by default.

```
apic ip-security-enabled:update --mode portaladmin [flags]
```

Options

```
      --enabled string                Set to true or false to enable or disable IP security (required)
      --format string                 Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                          Help for ip-security-enabled:update
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                 Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic licenses

Review the license for API Connect

Synopsis

Review the license for API Connect

```
apic licenses --mode portaladmin [flags]
```

Options

```
  -h, --help              Help for licenses
      --non-ibm-license   Display the non IBM license files.
      --notices           Display the notices file
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic modules

Modules operations

Synopsis

Modules operations

```
apic modules --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --core             Filter out extensions that are not in drupal core.
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for modules
      --no-core          Filter out extensions that are provided by drupal core.
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --package string   Filter by project packages. You can use multiple comma separated values.
  -s, --server string    management server endpoint (required)
      --status string    Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic modules:delete

Deletes one or more modules.

Synopsis

Deletes one or more modules. The provided list of modules must be comma separated.

```
apic modules:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for modules:delete
      --modules string   A list of modules separated by a comma. (required)
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic modules:disable

Disable one or more modules and their dependent modules.

Synopsis

Disable one or more modules and their dependent modules. The provided list of modules must be comma separated.

```
apic modules:disable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for modules:disable
      --modules string   A list of modules separated by a comma. (required)
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic modules:enable

Enable one or more module.

Synopsis

Enable one or more module by providing a comma separated list.

```
apic modules:enable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string         Name or ID of the catalog that the site belongs to. (required)
      --format string          Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                   Help for modules:enable
```

```
      --modules string          A list of modules separated by a comma. (required)
  -o, --org string              Name or ID of the organization that the catalog belongs to. (required)
      --resolve-dependencies    Attempt to download any missing dependencies. At the moment, only works when the module name is
the same as the project name.
  -s, --server string           management server endpoint (required)
      --skip                    Skip automatic downloading of libraries (c.f. devel).
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic modules:list

Shows a list of available modules.

Synopsis

Show a list of modules for the portal of the provided org and catalog.

```
apic modules:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --core             Filter out extensions that are not in drupal core.
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for modules:list
      --no-core          Filter out extensions that are provided by drupal core.
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --package string   Filter by project packages. You can use multiple comma separated values.
  -s, --server string    management server endpoint (required)
      --status string    Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic php-memory

Php Memory operations

Synopsis

Php Memory operations

```
apic php-memory --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for php-memory
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic php-memory:list

List the PHP memory limit on the platform.

Synopsis

Lists the PHP memory_limit setting on the portal platform.

```
apic php-memory:list --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for php-memory:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic php-memory:update

Set the PHP memory limit on the portal platform.

Synopsis

Sets the PHP memory limit on the portal platform for both admin and web containers

```
apic php-memory:update --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for php-memory:update
      --memory_value string          The memory value in megabytes (mb) (required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic platforms

Platforms operations

Synopsis

Platforms operations

```
apic platforms --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for platforms
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic platforms:list

List the platforms present on the portal service.

Synopsis

Lists all the platforms that are currently present on the Portal service.

```
apic platforms:list --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for platforms:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product

Product operations

Synopsis

Product operations

```
apic product --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for product
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:add-attachment

Add an attachment for the given product.

Synopsis

Add an attachment for a product within the developer portal of the provided org and catalog. A product can hold 10 attachments. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:add-attachment --mode portaladmin [flags]
```

Options

```
      --attachment_description string   A description of the attachment to be displayed to users.
      --attachment_name string          The name given to the attachment once it has been uploaded e.g. my-file.txt (required)
  -c, --catalog string                  Name or ID of the catalog that the site belongs to. (required)
      --format string                   Output format. One of [json yaml go-template=... go-template-file=...], defaults to
yaml.
  -h, --help                            Help for product:add-attachment
  -o, --org string                      Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string                   management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:add-tag

Add a tag for the given product.

Synopsis

Add a tag for a product within the developer portal of the provided org and catalog. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:add-tag --mode portaladmin [flags]
```

Options

```
  -c, --catalog string    Name or ID of the catalog that the site belongs to. (required)
      --format string     Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help              Help for product:add-tag
  -o, --org string        Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string     management server endpoint (required)
      --tag_name string   The tag name e.g. top_level_element/next_level_element (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:get

Get a specific product from a developer portal.

Synopsis

Get a specific product from the developer portal of the provided org and catalog. The id or name:version of a specific api needs to be provided. e.g. 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic product:get --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for product:get
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:get-document

Get a specific entire product document from a portal.

Synopsis

Get a specific entire product document from the portal of the provided organization and catalog. The id or name:version of a specific api needs to be provided. For example, 'id-of-api-called-example-3' or 'example:3.0.0'.

```
apic product:get-document --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for product:get-document
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:list

Shows a list of products from a site.

Synopsis

Show a list of products of the provided org and catalog.

```
apic product:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for product:list
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic product:set-icon

Set an icon image for the given product.

Synopsis

Set an icon image for a product within the developer portal of the provided org and catalog. The existing icon will be overwritten. The id or name:version of a specific product needs to be provided. e.g. 'id-of-product-called-example-3' or 'example:3.0.0'.

```
apic product:set-icon --mode portaladmin [flags]
```

Options

```
  -c, --catalog string            Name or ID of the catalog that the site belongs to. (required)
      --format string             Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                      Help for product:set-icon
      --icon_description string   A description of the icon to be displayed to users. Used as alt text for the image.
(required)
  -o, --org string                Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string             management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic security

Security operations

Synopsis

Security operations

```
apic security --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for security
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic security:clear-bans

Clear all user/IP bans for a site.

Synopsis

Clear any existing user/IP bans.

```
apic security:clear-bans --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for security:clear-bans
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic service

Service operations

Synopsis

Service operations

```
apic service --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic service:list

List the Portal service.

Synopsis

Lists the Portal service currently installed.

```
apic service:list --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic service-ip-allowlist

Service IP Allowlist operations.

Synopsis

Service IP Allowlist operations. The allowlist service enables specific IPs to be exempt from being blocked by portal security checks, for example, load balancer and proxy IPs.

```
apic service-ip-allowlist --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service-ip-allowlist
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license         Accept the license for API Connect
      --debug                  Enable debug output
      --debug-output string    Write debug output to file
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic service-ip-allowlist:add

Add IP(s) to the allowlist.

Synopsis

Add specific IP(s) to the allowlist.

```
apic service-ip-allowlist:add --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service-ip-allowlist:add
      --ips string                   Comma separated list of IPs (required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic service-ip-allowlist:delete

Delete all IPs from the allowlist.

Synopsis

Delete all of the IPs from the allowlist.

```
apic service-ip-allowlist:delete --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service-ip-allowlist:delete
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic service-ip-allowlist:list

List IP(s) currently on the allowlist.

Synopsis

List all of the IP(s) that are currently on the allowlist.

```
apic service-ip-allowlist:list --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service-ip-allowlist:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic service-ip-allowlist:remove

Remove specific IP(s) from the allowlist.

Synopsis

Remove one or more specific IP(s) from the allowlist.

```
apic service-ip-allowlist:remove --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for service-ip-allowlist:remove
      --ips string                   Comma separated list of IPs (required)
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                Management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site

Site operations

Synopsis

Site operations

```
apic site --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for site
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:cache-rebuild

Rebuilds the cache of a specific portal site.

Synopsis

Clears the existing cache and rebuilds a new one for a specific portal site.

```
apic site:cache-rebuild --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:cache-rebuild
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:check

Run a platform check against a portal site

Synopsis

Performs filesystem, database and API Manager checks against a portal site. This command is useful when trying to identify platform related problems if you are having issues with your portal site.

```
apic site:check --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:check
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:create-export

Create a task that exports an archive that contains the entire make-up of a site.

Synopsis

You can create a task to export an archive of the entire make-up of a site. You can then use that archive to quickly and simply override another site's content and configuration to make them identical.

```
apic site:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:create-export
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:create-import

Create a task that imports an archive that contains the make-up of a site.

Synopsis

You can create a task to import an archive of the make-up of a site. You can use the imported archive to quickly and simply override the site config, content and custom modules and themes for the specified site.

```
apic site:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:create-import
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:delete-export

If running, cancels a site export task.

Synopsis

If running, cancels the site export task and deletes any related artifacts that have been generated.

```
apic site:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:delete-import

If running, cancels a site import task.

Synopsis

If running, cancels the site import task and deletes any related artifacts that have been generated.

```
apic site:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic site:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:get-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license   Accept the license for API Connect
      --debug            Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:get-export

When ready, streams a site export artifact.

Synopsis

When ready, streams a site export artifact back to the related task id that is provided in the arguments.

```
apic site:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:get-import-status

Get the result of the site import task.

Synopsis

Get the result of the site import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic site:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml octet-stream go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:get-import-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:login-link

Gets an admin login link for a specific Portal site.

Synopsis

Returns a one time login link for the Drupal admin account for the site specified by org and catalog parameters

```
apic site:login-link --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:login-link
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site:state

Obtain the current state of a portal site.

Synopsis

Returns the state of a portal site. This command is useful as a general health check analysis of the portal site.

```
apic site:state --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site:state
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config

Site Config operations

Synopsis

Site Config operations

```
apic site-config --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for site-config
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:create-export

Create a task that exports an archive that contains the configuration of a site.

Synopsis

You can create a task to export an archive of the configuration of a site. You can then use that archive to quickly and simply override the configuration on another site.

```
apic site-config:create-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:create-export
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:create-import

Create a task that imports an archive that contains the configuration of a site.

Synopsis

You can create a task to import an archive of the configuration of a site. You can use the imported archive to quickly and simply override the configuration for the specified site.

```
apic site-config:create-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:create-import
      --no-poll          Do not poll the created task and just return the task ID
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:delete-export

If running, cancels a site configuration export task.

Synopsis

If running, cancels the site configuration export task and deletes any related artifacts that have been generated.

```
apic site-config:delete-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:delete-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:delete-import

Cancels a running site configuration import task.

Synopsis

Cancels the site configuration import task and deletes any generated artifacts.

```
apic site-config:delete-import --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:delete-import
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:get-export

When ready, streams a site configuration export artifact.

Synopsis

When ready, streams a site configuration export artifact back to the related task id that is provided in the arguments.

```
apic site-config:get-export --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:get-export
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:get-export-status

Returns information about the status of this task.

Synopsis

Returns information related to the task id that is provided in the arguments.

```
apic site-config:get-export-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:get-export-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
```

```
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic site-config:get-import-status

Get the result of the site configuration import task.

Synopsis

Get the result of the site configuration import task. If the import task has completed on the portal system, this command returns the result of the command. If the import task has not completed on the portal system, it returns the current status of the task.

```
apic site-config:get-import-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for site-config:get-import-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --task_id string   ID of the task created on the queue. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic sites

Sites operations

Synopsis

Sites operations

```
apic sites --mode portaladmin [flags]
```

Options

```
      --format string                Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                         Help for sites
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic sites:check

Run a platform check against all portal sites present on a portal service.

Synopsis

Performs filesystem, database and API Manager checks against all portal sites on a given portal service. This command is useful when trying to identify platform related problems on one of more of your portal sites.

```
apic sites:check --mode portaladmin [flags]
```

Options

```
      --format string   Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help            Help for sites:check
```

```
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic sites:list

List the sites present on the Portal service.

Synopsis

Lists all the sites that are currently present on the Portal service.

```
apic sites:list --mode portaladmin [flags]
```

Options

```
      --format string               Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                        Help for sites:list
      --portal_service_name string   The name of the portal service (required)
  -s, --server string                management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic themes

Themes operations

Synopsis

Themes operations

```
apic themes --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --core             Filter out extensions that are not in drupal core.
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for themes
      --no-core          Filter out extensions that are provided by drupal core.
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --package string   Filter by project packages. You can use multiple comma separated values.
  -s, --server string    management server endpoint (required)
      --status string    Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic themes:delete

Deletes one or more themes.

Synopsis

Deletes one or more themes. The provided list of themes must be comma separated.

```
apic themes:delete --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for themes:delete
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --themes string    A list of themes separated by a comma. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic themes:disable

Disable one or more themes.

Synopsis

Disable one or more themes and their dependent modules. The provided list of themes must be comma separated.

```
apic themes:disable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for themes:disable
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
      --themes string    A list of themes separated by a comma. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic themes:enable

Enable one or more theme.

Synopsis

Enable one or more theme by providing a comma separated list.

```
apic themes:enable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string          Name or ID of the catalog that the site belongs to. (required)
      --format string           Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help                    Help for themes:enable
  -o, --org string              Name or ID of the organization that the catalog belongs to. (required)
      --resolve-dependencies    Attempt to download any missing dependencies. At the moment, only works when the module name is
the same as the project name.
  -s, --server string           management server endpoint (required)
      --skip                    Skip automatic downloading of libraries (c.f. devel).
      --themes string           A list of themes separated by a comma. (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
```

```
      --live-help              Enable or disable tracking of limited usage information
  -m, --mode string            Toolkit operation mode (default "apim")
```

apic themes:list

Shows a list of available themes.

Synopsis

Show a list of themes for the portal of the provided org and catalog.

```
apic themes:list --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --core             Filter out extensions that are not in drupal core.
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for themes:list
      --no-core          Filter out extensions that are provided by drupal core.
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
      --package string   Filter by project packages. You can use multiple comma separated values.
  -s, --server string    management server endpoint (required)
      --status string    Filter by extension status. Choices _ enabled, disabled and/or 'not installed'. You can use multiple
comma separated values.
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic themes:set-default

Set a portal's default theme.

Synopsis

Set a portal's default theme. The theme provided must be one that is already enabled.

```
apic themes:set-default --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for themes:set-default
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic twig

Twig operations.

Synopsis

Twig operations.

```
apic twig --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for twig
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic twig:debug-disable

Disable twig debugging on a specific portal site.

Synopsis

Disable twig debugging on a specific portal site. Twig debugging helps locate twig template issues.

```
apic twig:debug-disable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for twig:debug-disable
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic twig:debug-enable

Enables twig debugging on a specific portal site.

Synopsis

Enables twig debugging on a specific portal site. Twig debugging helps locate twig template issues.

```
apic twig:debug-enable --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for twig:debug-enable
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic twig:debug-status

Shows the current state of twig debugging on a specific portal site.

Synopsis

View the current state of twig debugging on a specific portal site, if you are unsure whether it is enabled or disabled.

```
apic twig:debug-status --mode portaladmin [flags]
```

Options

```
  -c, --catalog string   Name or ID of the catalog that the site belongs to. (required)
      --format string    Output format. One of [json yaml go-template=... go-template-file=...], defaults to yaml.
  -h, --help             Help for twig:debug-status
  -o, --org string       Name or ID of the organization that the catalog belongs to. (required)
  -s, --server string    management server endpoint (required)
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

apic version

Get the APIConnect toolkit version

Synopsis

Get the APIConnect toolkit version

```
apic version --mode portaladmin [flags]
```

Options

```
  -h, --help   Help for version
```

Options inherited from parent commands

```
      --accept-license        Accept the license for API Connect
      --debug                 Enable debug output
      --debug-output string   Write debug output to file
      --live-help             Enable or disable tracking of limited usage information
  -m, --mode string           Toolkit operation mode (default "apim")
```

API Connect REST APIs

The API Connect platform REST APIs provide complete access to the capability of the platform.

The API Connect platform REST APIs can be used for the following actions:

- Automate the administration of the platform.
- Implement scripts and tools to support a continuous integration environment for API development and publishing.
- Manage catalogs of APIs, and their subscribers.

Note: To authenticate with the API Connect Platform REST APIs, generate an API Key and then exchange it for an access token to use when calling the APIs. You can exchange it for an access token with the following example:

```
curl -v -k -X POST -d '{"api_key": "****", "client_id": "client-id", "client_secret": "client-secret", "grant_type":
"api_key"}' -H 'Content-Type: application/json' -H 'Accept: application/json' https://platform-api.
{{region}}.apiconnect.automation.ibm.com/api/token
```

The operations provided in the REST APIs also correspond directly with the commands in the developer toolkit command-line tool (CLI).

You can use the URL to navigate directly to specific APIs and operations in the OpenAPI Explorer Documentation for the API Connect REST APIs. You can also select the APIs for a specific version by using the drop-down menu in the header bar.

For descriptions of the platform APIs, see the API Connect REST API documentation.

Example: Using the platform REST APIs to publish a product containing a SOAP API

Review this sample scenario to see how you can use the API Connect platform REST APIs to publish a product that contains a SOAP API.

This example demonstrates how to format a request to publish the product, and includes sample files that illustrate the product definition, the API definition, and the SOAP API's WSDL description.

Using the `Catalogs` API

Publish the product using the API Connect platform **`Catalogs`** API:

- To publish a product to a catalog, use the following syntax:

`POST catalogs/{org}/{catalog}/publish`

- To publish a product to a space within a catalog, additionally specify the space with the following syntax:

  `POST catalogs/{org}/{catalog}/{space}/publish`

The payload in both cases is a multipart, form-encoded body containing three required parts and one optional part:

- (Required) A part named **product** with media type **application/yaml** or **application/json**, containing the yaml or JSON source for the product to be published.
- (Required) One or more occurrences of a part named **openapi** with media type **application/yaml** or **application/json**, containing the YAML or JSON source for each API in the product.
- (Required for SOAP APIs) One or more occurrences of a part named **wsdl** with media type **application/wsdl**, **application/wsdl+xml**, **text/xml**, or **application/zip** containing the WSDL definitions (and referenced XSD files if applicable) associated with the API content.
- (Optional) A part named **gateway_service_urls** with media type **application/yaml** or **application/json** which is a YAML or JSON array of URLs to specify a subset of the configured gateway services within the catalog or space as publish targets. These URLs can be obtained using the following calls:
  - For a catalog:

    `GET /api/catalogs/{org}/{catalog}/configured-gateway-services`

  - For a space:

    `GET /spaces/{org}/{catalog}/{space}/configured-gateway-services`

Within the **product** content, the form of the references to APIs in the **apis** section is significant. Each API reference must use the **name** property instead of the **$ref** form that may appear in other situations. Here is an example showing a product that refers to the API **globalweather** version **1.0.0** in the correct form:

```
info:
  version: 1.0.0
  title: GlobalWeatherProduct
  name: globalweatherproduct
plans:
  ...
apis:
  globalweather1.0.0:
    name: globalweather:1.0.0
...
```

Example publish operation using cURL

This example publishes a product containing a SOAP API to the *Sandbox* catalog of the *onlinebanking* provider organization, using the **curl -F** option to create a set of form-encoded payload parts and the **@** prefix to read the content for each part from a file. Note the part name (**product**, **openapi**, **wsdl**) and the content type of each part given by the **type=** value.

In this example, the access token has been retrieved earlier and placed in an environment variable named **TOKEN**. The command is split onto multiple lines for ease of reading but must be entered as a single line.

Request:

```
curl -v -k
  -F "product=@globalweatherproduct_1.0.0.yaml;type=application/yaml"
  -F "openapi=@globalweather_1.0.0.yaml;type=application/yaml"
  -F "wsdl=@globalweather.wsdl;type=application/wsdl"
  -H "Authorization: Bearer $TOKEN"
  -H 'Content-Type: multipart/form-data'
  -H 'Accept: */*' https://dev.apic.example.com/api/catalogs/onlinebanking/sandbox/publish
```

Response:

```
201 Created

{
    "type": "product",
    "api_version": "2.0.0",
    "id": "d03129a4-9a99-4c88-85e6-d51bbf312164",
    "name": "globalweatherproduct",
    "version": "1.0.0",
    "title": "GlobalWeatherProduct",
    "state": "published",
    "scope": "catalog",
    "gateway_types": [
        "datapower-api-gateway"
    ],
    "billing_urls": [],
    "api_urls": [
        "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-
96a8ca9667fd/apis/d5986b78-d556-4457-8c38-951c471f57fc"
    ],
    "gateway_service_urls": [
        "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-
96a8ca9667fd/configured-gateway-services/a416f39d-39e1-484c-9f7e-4a599abefec8"
    ],
    "oauth_provider_urls": [],
    "plans": [
        {
            "title": "Default Plan",
            "name": "default-plan",
            "apis": [
                {
                    "id": "d5986b78-d556-4457-8c38-951c471f57fc",
                    "name": "globalweather",
                    "title": "GlobalWeather",
```

```
                    "version": "1.0.0",
                    "url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-
93e8-96a8ca9667fd/apis/d5986b78-d556-4457-8c38-951c471f57fc"
                }
            ]
        }
    ],
    "visibility": {
        "view": {
            "type": "public",
            "enabled": true
        },
        "subscribe": {
            "type": "authenticated",
            "enabled": true
        }
    },
    "task_urls": [],
    "created_at": "2021-12-15T07:35:40.295Z",
    "updated_at": "2021-12-15T07:35:40.295Z",
    "org_url": "https://dev.apic.example.com/api/orgs/87227774-d297-4064-8908-b8f0b1db71a5",
    "catalog_url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-
96a8ca9667fd",
    "url": "https://dev.apic.example.com/api/catalogs/87227774-d297-4064-8908-b8f0b1db71a5/1bc8f4ba-4178-4a4b-93e8-
96a8ca9667fd/products/d03129a4-9a99-4c88-85e6-d51bbf312164"
```

The request specifies three files, which are included with this example for reference:

- Product definition `-F "product=@globalweatherproduct_1.0.0.yaml;type=application/yaml"`
- API definition `-F "openapi=@globalweather_1.0.0.yaml;type=application/yaml"`
- WSDL description `-F "wsdl=@globalweather.wsdl;type=application/wsdl"`

Example Product file: globalweatherproduct_1.0.0.yaml

The following code sample contains a Product definition.

The Product definition file globalweatherproduct_1.0.0.yaml is referenced by the example scenario in Example: Using the platform REST APIs to publish a product containing a SOAP API.

```
info:
  version: 1.0.0
  title: GlobalWeatherProduct
  name: globalweatherproduct
gateways:
  - datapower-api-gateway
plans:
  default-plan:
    title: Default Plan
    description: Default Plan
    rate-limits:
      default:
        value: 100/1hour
    apis:
      globalweather1.0.0: {}
apis:
  globalweather1.0.0:
    name: globalweather:1.0.0
visibility:
  view:
    type: public
    orgs: []
    tags: []
    enabled: true
  subscribe:
    type: authenticated
    orgs: []
    tags: []
    enabled: true
product: 1.0.0
```

Example API definition: globalweather_1.0.0.yaml

The following code sample contains an API definition.

The API definition file globalweather_1.0.0.yaml is referenced by the example scenario in Example: Using the platform REST APIs to publish a product containing a SOAP API.

```
swagger: '2.0'
info:
  title: GlobalWeather
  description: ''
  x-ibm-name: globalweather
  version: 1.0.0
schemes:
  - https
basePath: /globalweather
produces:
```

```
  - application/xml
consumes:
  - text/xml
securityDefinitions:
  clientID:
    type: apiKey
    in: header
    name: X-IBM-Client-Id
security:
  - clientID: []
x-ibm-configuration:
  type: wsdl
  phase: realized
  enforced: true
  testable: true
  gateway: datapower-api-gateway
  cors:
    enabled: true
  wsdl-definition:
    wsdl: globalweather.wsdl
    service: GlobalWeather
    port: GlobalWeatherSoap
    soap-version: '1.1'
  assembly:
    execute:
      - invoke:
          title: invoke
          target-url: 'http://www.webservicex.com/globalweather.asmx'
          version: 2.0.0
          header-control:
            type: blocklist
            values: []
          parameter-control:
            type: blocklist
            values: []
  x-ibm-apiconnect-wsdl:
    package-version: 2.0.35
    options: {}
    messages:
      info:
        - message: >-
            The wsdl 'service' has multiple 'ports'. The api is generated using
            information in the first soap 'port'.
      warning: []
      error: []
paths:
  /GetWeather:
    post:
      summary: Operation GetWeather
      description: Get weather report for all major cities around the world.
      operationId: GetWeather
      x-ibm-soap:
        soap-action: 'http://www.webserviceX.NET/GetWeather'
        soap-operation: '{http://www.webserviceX.NET}GetWeather'
      parameters:
        - in: body
          name: body
          required: true
          schema:
            $ref: '#/definitions/GetWeatherInput'
      responses:
        default:
          description: ''
          schema:
            $ref: '#/definitions/GetWeatherOutput'
  /GetCitiesByCountry:
    post:
      summary: Operation GetCitiesByCountry
      description: "Get all major\n\t\t\t\tcities by country name(full / part)."
      operationId: GetCitiesByCountry
      x-ibm-soap:
        soap-action: 'http://www.webserviceX.NET/GetCitiesByCountry'
        soap-operation: '{http://www.webserviceX.NET}GetCitiesByCountry'
      parameters:
        - in: body
          name: body
          required: true
          schema:
            $ref: '#/definitions/GetCitiesByCountryInput'
      responses:
        default:
          description: ''
          schema:
            $ref: '#/definitions/GetCitiesByCountryOutput'
definitions:
  Security:
    xml:
      namespace: >-
        http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
      prefix: wsse
    description: Header for WS-Security
    type: object
    properties:
      UsernameToken:
        xml:
          namespace: >-
```

```
          http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
        prefix: wsse
      type: object
      properties:
        Username:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
            prefix: wsse
          type: string
        Password:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
            prefix: wsse
          type: string
        Nonce:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
            prefix: wsse
          type: string
          properties:
            EncodingType:
              xml:
                namespace: ''
                attribute: true
              type: string
        Created:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
            prefix: wsu
          type: string
    Timestamp:
      xml:
        namespace: >-
          http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
        prefix: wsu
      type: object
      properties:
        Created:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
            prefix: wsu
          type: string
        Expires:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
            prefix: wsu
          type: string
        Id:
          xml:
            namespace: >-
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
            prefix: wsu
            attribute: true
          type: string
GetWeatherInput:
  description: Input message for wsdl operation GetWeather
  type: object
  properties:
    Envelope:
      xml:
        namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
        prefix: soapenv
      type: object
      properties:
        Header:
          $ref: '#/definitions/GetWeatherHeader'
        Body:
          xml:
            namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
            prefix: soapenv
          type: object
          properties:
            GetWeather:
              $ref: '#/definitions/GetWeather_element_tns'
          required:
            - GetWeather
      required:
        - Body
  required:
    - Envelope
  x-ibm-schema:
    wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
    wsdl-operation: GetWeather
    wsdl-message-direction-or-name: GetWeatherRequest
  example: >-

    <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
     <soapenv:Header>
      <!-- The Security element should be removed if WS-Security is not enabled on the SOAP target-url -->
```

```
       <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:UsernameToken>
         <wsse:Username>string</wsse:Username>
         <wsse:Password>string</wsse:Password>
         <wsse:Nonce EncodingType="string">string</wsse:Nonce>
         <wsu:Created>string</wsu:Created>
        </wsse:UsernameToken>
        <wsu:Timestamp wsu:Id="string">
         <wsu:Created>string</wsu:Created>
         <wsu:Expires>string</wsu:Expires>
        </wsu:Timestamp>
       </wsse:Security>
      </soapenv:Header>
      <soapenv:Body>
       <tns:GetWeather xmlns:tns="http://www.webserviceX.NET"><!-- mandatory -->
        <tns:CityName>string</tns:CityName>
        <tns:CountryName>string</tns:CountryName>
       </tns:GetWeather>
      </soapenv:Body>
     </soapenv:Envelope>
  GetWeatherHeader:
    description: Input headers for wsdl operation GetWeather
    type: object
    properties:
      Security:
        $ref: '#/definitions/Security'
  GetWeatherOutput:
    description: Output message for wsdl operation GetWeather
    type: object
    properties:
      Envelope:
        xml:
          namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
          prefix: soapenv
        type: object
        properties:
          Body:
            xml:
              namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
              prefix: soapenv
            type: object
            properties:
              GetWeatherResponse:
                $ref: '#/definitions/GetWeatherResponse_element_tns'
        required:
          - Body
    required:
      - Envelope
    x-ibm-schema:
      wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
      wsdl-operation: GetWeather
      wsdl-message-direction-or-name: GetWeatherResponse
    example: >-

      <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
       <soapenv:Body>
        <tns:GetWeatherResponse xmlns:tns="http://www.webserviceX.NET">
         <tns:GetWeatherResult>string</tns:GetWeatherResult>
        </tns:GetWeatherResponse>
       </soapenv:Body>
      </soapenv:Envelope>
  GetCitiesByCountryInput:
    description: Input message for wsdl operation GetCitiesByCountry
    type: object
    properties:
      Envelope:
        xml:
          namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
          prefix: soapenv
        type: object
        properties:
          Header:
            $ref: '#/definitions/GetCitiesByCountryHeader'
          Body:
            xml:
              namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
              prefix: soapenv
            type: object
            properties:
              GetCitiesByCountry:
                $ref: '#/definitions/GetCitiesByCountry_element_tns'
            required:
              - GetCitiesByCountry
        required:
          - Body
    required:
      - Envelope
    x-ibm-schema:
      wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
      wsdl-operation: GetCitiesByCountry
      wsdl-message-direction-or-name: GetCitiesByCountryRequest
    example: >-

      <soapenv:Envelope
```

```
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
       <soapenv:Header>
        <!-- The Security element should be removed if WS-Security is not enabled on the SOAP target-url -->
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <wsse:UsernameToken>
          <wsse:Username>string</wsse:Username>
          <wsse:Password>string</wsse:Password>
          <wsse:Nonce EncodingType="string">string</wsse:Nonce>
          <wsu:Created>string</wsu:Created>
         </wsse:UsernameToken>
         <wsu:Timestamp wsu:Id="string">
          <wsu:Created>string</wsu:Created>
          <wsu:Expires>string</wsu:Expires>
         </wsu:Timestamp>
        </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
        <tns:GetCitiesByCountry xmlns:tns="http://www.webserviceX.NET"><!-- mandatory -->
         <tns:CountryName>string</tns:CountryName>
        </tns:GetCitiesByCountry>
       </soapenv:Body>
      </soapenv:Envelope>
  GetCitiesByCountryHeader:
    description: Input headers for wsdl operation GetCitiesByCountry
    type: object
    properties:
      Security:
        $ref: '#/definitions/Security'
  GetCitiesByCountryOutput:
    description: Output message for wsdl operation GetCitiesByCountry
    type: object
    properties:
      Envelope:
        xml:
          namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
          prefix: soapenv
        type: object
        properties:
          Body:
            xml:
              namespace: 'http://schemas.xmlsoap.org/soap/envelope/'
              prefix: soapenv
            type: object
            properties:
              GetCitiesByCountryResponse:
                $ref: '#/definitions/GetCitiesByCountryResponse_element_tns'
        required:
          - Body
    required:
      - Envelope
    x-ibm-schema:
      wsdl-port: '{http://www.webserviceX.NET}GlobalWeatherSoap'
      wsdl-operation: GetCitiesByCountry
      wsdl-message-direction-or-name: GetCitiesByCountryResponse
    example: >-

      <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
       <soapenv:Body>
        <tns:GetCitiesByCountryResponse xmlns:tns="http://www.webserviceX.NET">
         <tns:GetCitiesByCountryResult>string</tns:GetCitiesByCountryResult>
        </tns:GetCitiesByCountryResponse>
       </soapenv:Body>
      </soapenv:Envelope>
  GetWeather_element_tns:
    xml:
      namespace: 'http://www.webserviceX.NET'
      prefix: tns
      name: GetWeather
    type: object
    properties:
      CityName:
        xml:
          namespace: 'http://www.webserviceX.NET'
          prefix: tns
        type: string
      CountryName:
        xml:
          namespace: 'http://www.webserviceX.NET'
          prefix: tns
        type: string
    example: |-

      <tns:GetWeather xmlns:tns="http://www.webserviceX.NET">
       <tns:CityName>string</tns:CityName>
       <tns:CountryName>string</tns:CountryName>
      </tns:GetWeather>
  GetWeatherResponse_element_tns:
    xml:
      namespace: 'http://www.webserviceX.NET'
      prefix: tns
      name: GetWeatherResponse
    type: object
    properties:
      GetWeatherResult:
```

```
        xml:
          namespace: 'http://www.webserviceX.NET'
          prefix: tns
        type: string
    example: |-

      <tns:GetWeatherResponse xmlns:tns="http://www.webserviceX.NET">
       <tns:GetWeatherResult>string</tns:GetWeatherResult>
      </tns:GetWeatherResponse>
  GetCitiesByCountry_element_tns:
    xml:
      namespace: 'http://www.webserviceX.NET'
      prefix: tns
      name: GetCitiesByCountry
    type: object
    properties:
      CountryName:
        xml:
          namespace: 'http://www.webserviceX.NET'
          prefix: tns
        type: string
    example: |-

      <tns:GetCitiesByCountry xmlns:tns="http://www.webserviceX.NET">
       <tns:CountryName>string</tns:CountryName>
      </tns:GetCitiesByCountry>
  GetCitiesByCountryResponse_element_tns:
    xml:
      namespace: 'http://www.webserviceX.NET'
      prefix: tns
      name: GetCitiesByCountryResponse
    type: object
    properties:
      GetCitiesByCountryResult:
        xml:
          namespace: 'http://www.webserviceX.NET'
          prefix: tns
        type: string
    example: |-

      <tns:GetCitiesByCountryResponse xmlns:tns="http://www.webserviceX.NET">
       <tns:GetCitiesByCountryResult>string</tns:GetCitiesByCountryResult>
      </tns:GetCitiesByCountryResponse>
```

Example WSDL file: globalweather.wsdl

The following code sample contains a WSDL description.

The WSDL description file globalweather.wsdl is referenced by the example scenario in Example: Using the platform REST APIs to publish a product containing a SOAP API.

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        xmlns:tns="http://www.webserviceX.NET" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace="http://www.webserviceX.NET">
        <wsdl:types>
                <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET">
                        <s:element name="GetWeather">
                                <s:complexType>
                                        <s:sequence>
                                                <s:element minOccurs="0" maxOccurs="1" name="CityName"
                                                        type="s:string" />
                                                <s:element minOccurs="0" maxOccurs="1" name="CountryName"
                                                        type="s:string" />
                                        </s:sequence>
                                </s:complexType>
                        </s:element>
                        <s:element name="GetWeatherResponse">
                                <s:complexType>
                                        <s:sequence>
                                                <s:element minOccurs="0" maxOccurs="1" name="GetWeatherResult"
                                                        type="s:string" />
                                        </s:sequence>
                                </s:complexType>
                        </s:element>
                        <s:element name="GetCitiesByCountry">
                                <s:complexType>
                                        <s:sequence>
                                                <s:element minOccurs="0" maxOccurs="1" name="CountryName"
                                                        type="s:string" />
                                        </s:sequence>
                                </s:complexType>
                        </s:element>
                        <s:element name="GetCitiesByCountryResponse">
                                <s:complexType>
                                        <s:sequence>
                                                <s:element minOccurs="0" maxOccurs="1"
                                                        name="GetCitiesByCountryResult" type="s:string" />
                                        </s:sequence>
```

```
                        </s:complexType>
                </s:element>
                <s:element name="string" nillable="true" type="s:string" />
        </s:schema>
</wsdl:types>
<wsdl:message name="GetWeatherSoapIn">
        <wsdl:part name="parameters" element="tns:GetWeather" />
</wsdl:message>
<wsdl:message name="GetWeatherSoapOut">
        <wsdl:part name="parameters" element="tns:GetWeatherResponse" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountrySoapIn">
        <wsdl:part name="parameters" element="tns:GetCitiesByCountry" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountrySoapOut">
        <wsdl:part name="parameters" element="tns:GetCitiesByCountryResponse" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpGetIn">
        <wsdl:part name="CityName" type="s:string" />
        <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpGetOut">
        <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpGetIn">
        <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpGetOut">
        <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpPostIn">
        <wsdl:part name="CityName" type="s:string" />
        <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetWeatherHttpPostOut">
        <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpPostIn">
        <wsdl:part name="CountryName" type="s:string" />
</wsdl:message>
<wsdl:message name="GetCitiesByCountryHttpPostOut">
        <wsdl:part name="Body" element="tns:string" />
</wsdl:message>
<wsdl:portType name="GlobalWeatherSoap">
        <wsdl:operation name="GetWeather">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
                        Get weather report for all major cities around the world.
                </wsdl:documentation>
                <wsdl:input message="tns:GetWeatherSoapIn" />
                <wsdl:output message="tns:GetWeatherSoapOut" />
        </wsdl:operation>
        <wsdl:operation name="GetCitiesByCountry">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
                        cities by country name(full / part).</wsdl:documentation>
                <wsdl:input message="tns:GetCitiesByCountrySoapIn" />
                <wsdl:output message="tns:GetCitiesByCountrySoapOut" />
        </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="GlobalWeatherHttpGet">
        <wsdl:operation name="GetWeather">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
                        Get weather report for all major cities around the world.
                </wsdl:documentation>
                <wsdl:input message="tns:GetWeatherHttpGetIn" />
                <wsdl:output message="tns:GetWeatherHttpGetOut" />
        </wsdl:operation>
        <wsdl:operation name="GetCitiesByCountry">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
                        cities by country name(full / part).</wsdl:documentation>
                <wsdl:input message="tns:GetCitiesByCountryHttpGetIn" />
                <wsdl:output message="tns:GetCitiesByCountryHttpGetOut" />
        </wsdl:operation>
</wsdl:portType>
<wsdl:portType name="GlobalWeatherHttpPost">
        <wsdl:operation name="GetWeather">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
                        Get weather report for all major cities around the world.
                </wsdl:documentation>
                <wsdl:input message="tns:GetWeatherHttpPostIn" />
                <wsdl:output message="tns:GetWeatherHttpPostOut" />
        </wsdl:operation>
        <wsdl:operation name="GetCitiesByCountry">
                <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Get all major
                        cities by country name(full / part).</wsdl:documentation>
                <wsdl:input message="tns:GetCitiesByCountryHttpPostIn" />
                <wsdl:output message="tns:GetCitiesByCountryHttpPostOut" />
        </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeatherSoap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="GetWeather">
                <soap:operation soapAction="http://www.webserviceX.NET/GetWeather"
                  />
                <wsdl:input>
                        <soap:body use="literal" />
                </wsdl:input>
```

```
                        <wsdl:output>
                                <soap:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="GetCitiesByCountry">
                        <soap:operation soapAction="http://www.webserviceX.NET/GetCitiesByCountry"
                                />
                        <wsdl:input>
                                <soap:body use="literal" />
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:binding name="GlobalWeatherSoap12" type="tns:GlobalWeatherSoap">
                <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
                <wsdl:operation name="GetWeather">
                        <soap12:operation soapAction="http://www.webserviceX.NET/GetWeather"
                                />
                        <wsdl:input>
                                <soap12:body use="literal" />
                        </wsdl:input>
                        <wsdl:output>
                                <soap12:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="GetCitiesByCountry">
                        <soap12:operation soapAction="http://www.webserviceX.NET/GetCitiesByCountry"
                                />
                        <wsdl:input>
                                <soap12:body use="literal" />
                        </wsdl:input>
                        <wsdl:output>
                                <soap12:body use="literal" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:binding name="GlobalWeatherHttpGet" type="tns:GlobalWeatherHttpGet">
                <http:binding verb="GET" />
                <wsdl:operation name="GetWeather">
                        <http:operation location="/GetWeather" />
                        <wsdl:input>
                                <http:urlEncoded />
                        </wsdl:input>
                        <wsdl:output>
                                <mime:mimeXml part="Body" />
                        </wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="GetCitiesByCountry">
                        <http:operation location="/GetCitiesByCountry" />
                        <wsdl:input>
                                <http:urlEncoded />
                        </wsdl:input>
                        <wsdl:output>
                                <mime:mimeXml part="Body" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:binding name="GlobalWeatherHttpPost" type="tns:GlobalWeatherHttpPost">
                <http:binding verb="POST" />
                <wsdl:operation name="GetWeather">
                        <http:operation location="/GetWeather" />
                        <wsdl:input>
                                <mime:content type="application/x-www-form-urlencoded" />
                        </wsdl:input>
                        <wsdl:output>
                                <mime:mimeXml part="Body" />
                        </wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="GetCitiesByCountry">
                        <http:operation location="/GetCitiesByCountry" />
                        <wsdl:input>
                                <mime:content type="application/x-www-form-urlencoded" />
                        </wsdl:input>
                        <wsdl:output>
                                <mime:mimeXml part="Body" />
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="GlobalWeather">
                <wsdl:port name="GlobalWeatherSoap" binding="tns:GlobalWeatherSoap">
                        <soap:address location="http://www.webservicex.com/globalweather.asmx" />
                </wsdl:port>
                <wsdl:port name="GlobalWeatherSoap12" binding="tns:GlobalWeatherSoap12">
                        <soap12:address location="http://www.webservicex.com/globalweather.asmx" />
                </wsdl:port>
                <wsdl:port name="GlobalWeatherHttpGet" binding="tns:GlobalWeatherHttpGet">
                        <http:address location="http://www.webservicex.com/globalweather.asmx" />
                </wsdl:port>
                <wsdl:port name="GlobalWeatherHttpPost" binding="tns:GlobalWeatherHttpPost">
                        <http:address location="http://www.webservicex.com/globalweather.asmx" />
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

API Connect TLS certificates

Details of all Kubernetes TLS certificate objects that are created and used in a default installation of API Connect.

Overview

TLS is used to secure communications between API Connect pods, subsystems, external services, and users. The TLS certificates that are used by API Connect are created automatically during the installation of API Connect. The certificates are managed by an open source product called cert-manager.

The certificates that are described in this topic all exist as Kubernetes certificate objects in the API Connect namespace where the subsystem that uses the certificate is installed. Run `kubectl get certs -n <namespace>` to see the certificates in your environment. Some of the API Connect Kubernetes certificates are also stored in the management subsystem database. These certificates can be seen in the Cloud Manager UI, in the truststores and keystores of the subsystem default TLS profiles. When the management subsystem `apim` pod is restarted, the certificates in the management database are reloaded from the Kubernetes certificates.

This topic does not cover additional TLS profiles and certificates that API Connect administrators and provider organization owners can create and configure in the Cloud Manager and API Manager UIs. These certificates are stored in the management subsystem database only and do not exist in the Kubernetes environment.
Note: The API invocation certificate that callers to the published APIs on your gateways see is configured in the Cloud Manager UI when you register your gateway services. For more information about API invocation certificates, see Registering a gateway service and TLS profiles overview.
The certificates that are created and used by API Connect are categorized as follows:

- CA certificates - API Connect creates self-signed CA certificates to sign all the API Connect end-entity certificates. The CA certificates are:
  - `ingress-ca` - used by the ingress-issuer. The ingress-ca signs all user-facing and inter-subsystem certificates.
  - `management-ca` - signs all the management intra-subsystem certificates.
  - `portal-ca` - signs all the portal intra-subsystem certificates.
  - `analytics-ca` - signs all the analytics intra-subsystem certificates.

  CA certificates have a default duration of 10 years. All the end-entity certificates have a default duration of 2 years.
- Ingress certificates - API Connect creates a self-signed issuer that is called the ingress-issuer. Ingress certificates are the API Connect certificates that are issued by the ingress-issuer. Ingress certificates are divided into two types:
  - Inter-subsystem certificates - certificates that are used for communication between API Connect subsystems, for example between the management subsystem and gateway. By default, communication between most subsystems uses mTLS. From v10.0.5.3, it is possible to use JWT instead of mTLS: Using JWT instead of mTLS on Kubernetes and OpenShift, Using JWT instead of mTLS on OVA.
  - User-facing certificates - certificates that API Connect users see when they interact with the API Connect UIs and REST API.
- Intra-subsystem certificates - certificates that are used between pods in the same subsystem. Intra-subsystem certificates are signed by a subsystem specific CA certificate: `management-ca`, `portal-ca`, `analytics-ca`. The gateway subsystem is a single pod, and has no intra-subsystem certificates.

On OVA deployments, certificate management is done with the `apicup` command, see API Connect TLS certificate best practices. Do not login to your VMs by `ssh` and attempt to modify certificates directly, unless advised by IBM.

Note: On Cloud Pak for Integration, and OpenShift top-level CR deployments, some certificate names are contracted and prefixed with the `APIConnectCluster` instance name. For example, the certificate `managment-ca` is called `<apic instance name>-mgmt-ca`.

Customizing certificates

It is possible to customize certificates with ones from your own organization. However it is recommended to customize only the user-facing certificates, which are the certificates that your users see when they access the API Connect UIs, CLI, and the REST API.

All other certificates are used internally between API Connect microservices.
Note: The inter and intra-subsystem certificates use of a self-signed root CA presents no security risk. For internal communications with a limited number of predefined endpoints, use of locally generated self-signed CA certificates can be more secure than external CA certificates, as certificates are only ever issued for the known components and the domain of trust is kept as small as possible.

Renewing certificates

Cert-manager handles the renewal of expired certificates so that the user does not need to monitor or manually renew any of the certificates. However, there are scenarios where the renewal of certificates must be triggered manually. Every API Connect certificate has a corresponding Kubernetes secret object of the same name, deleting this secret triggers cert-manager to re-create the secret with a new TLS certificate. For more information about certificate renewal, see Renewing TLS certificates.
Remember: The Kubernetes certificate objects do not contain the TLS certificates themselves, they are pointers to Kubernetes secrets that contain the TLS certificate. See Key Concepts: Cert-manager, Issuers, and Secrets.
For most of the certificates, when they are renewed, the pods that use those certificates are automatically restarted so that they pick up the updated certificate. The minority of certificates that require manual pod restarts after renewal are identified in the reference tables.

If you renew a CA certificate, any end-entity certificates that use this CA certificate must also be renewed. The following command identifies and renews all certificates that are signed by `ingress-ca` in the specified namespace:

```
kubectl get secrets -n <namespace> -o custom-columns='NAME:.metadata.name,ISSUER:.metadata.annotations.cert-manager\.io/issuer-name' --no-headers=true | grep ingress-issuer | awk '{ print $1 }' | xargs kubectl delete secret -n <namespace>
```

Warning: Cert-manager automatically renews the `ingress-ca` 30 days before it expires. The automatic renewal of `ingress-ca` does not trigger the renewal of the end-entity certificates that are signed by `ingress-ca`. When `ingress-ca` is automatically renewed, you must manually renew all end-entity certificates.

Reference tables

The tables show all the API Connect certificates that are created in all possible deployment configurations.

Each table corresponds to a subsystem, and shows the certificates that are used by that subsystem.

Note: On Cloud Pak for Integration or top-level CR deployments, some of the certificates have a shortened name. Where two names for a certificate are shown in the tables, the second name is the name that is used on Cloud Pak for Integration or top-level CR deployments. For example, **`management-ca or mgmt-ca`**.

Table 1. Management certificates

Certificate name	Issuer	Description
`-ingress-ca`	**`selfsigning-issuer`**	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem **`apim`** pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the **`portal-www`** pods for the update to take effect.
`management-ca or mgmt-ca`	**`selfsigning-issuer`**	The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using **`in-cluster`** communication. See In-cluster service communication between subsystems
`management-client or mgmt-client`	**`management-ca`**	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
`management-server or mgmt-server`	**`management-ca`**	Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: **`*.<namespace>`** **`*.<namespace>.svc`** **`*.<instance name>-server.<namespace>.svc`** **`<instance name>-server`**
`analytics-client-client or a7s-cl-client`	**`ingress-issuer`**	Legacy certificate from pre-v10.0.5 releases. Not used for anything and safe to delete. In v10.0.5, communication with the analytics subsystem uses the **`analytics-ingestion-client`** certificate.
`analytics-ingestion-client or a7s-ing-client`	**`ingress-issuer`**	Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have: • The same CA as the server certificate **`analytics-ai-endpoint`** on the analytics subsystem. • The common name must match the **`spec.clientSubjectDN`** in the analytics CR: **`kubectl describe cert analytics-admin-client`** **`Spec:`** **`  Common Name:  a7s-ing-client`** **`kubectl get a7s -o yaml`** **`  spec:`** **`    clientSubjectDN: CN=a7s-ing-client`** For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be **`CN=a7s-ingestion-client`**, or they could both be **`CN=a7s-ingestion-client, O=cert-manager`**, but they must be identical. To update this certificate, use kubectl. Restart the **`apim`**, **`taskmanager`**, and **`analytics-proxy`** pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.
`portal-admin-client or ptl-adm-client`	**`ingress-issuer`**	Client certificate used for communication with the portal subsystem on the **`portalAdminEndpoint`**. This certificate must have: • The same CA as the server certificate **`portal-admin or ptl-director`** on the portal subsystem. • The common name must match the **`spec.adminClientSubjectDN`** in the portal CR: **`kubectl describe cert portal-admin-client`** **`Spec:`** **`  Common Name:  ptl-adm-client`** **`kubectl get ptl -o yaml`** **`  spec:`** **`    adminClientSubjectDN: CN=ptl-adm-client`** For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be **`CN=portal-admin-client`**, or they could both be **`CN=ptl-adm-client, O=cert-manager`**, but they must be identical.
`gateway-client-client or gw-dr-client`	**`ingress-issuer`**	Client certificate for communications with the gateway service. Restart the **`apim`**, and **`taskmanager`** pods after update. This certificate must have the same CA as the **`gwv6-manager-endpoint or gw-gateway-manager`** certificate on the gateway.
`api-endpoint or mgmt-platform-api`	**`ingress-issuer`**	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.

Certificate name	Issuer	Description
`consumer-endpoint or mgmt-consumer-api`	`ingress-issuer`	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
`apim-endpoint or mgmt-api-manager`	`ingress-issuer`	API Manager UI server certificate.
`cm-endpoint or mgmt-admin`	`ingress-issuer`	Cloud Manager UI server certificate.
`db-client-apicuser`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `db-client-pgbouncer`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`db-client-postgres`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `db-client-primaryuser`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`postgres`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `postgres-pgbouncer`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `postgres-operator`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`natscluster-mgmt`	`management-ca`	Intra-subsystem certificate for the `nats` pods.
v10.0.6 only `db-client-replicator`	`management-ca`	2DCDR deployments only. Client certificate used by the `<management_CR>-tunnel` pod to connect to the other data center's `<management_CR>-tunnel` pod.
`mgmt-replication-client`	`ingress-issuer`	2DCDR deployments only. Client certificate used in the warm-standby data center's `<remote-sitename>-postgres` pod to connect to the active data center.
`mgmt-replication-server`	`ingress-issuer`	2DCDR deployments only. Server certificate used in the active data center's `<management_CR>-tunnel` pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the `spec.multiSiteHA.replicationEndpoint` defined in the management CR.
`hub-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `hub-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`turnstile-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `turnstile-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.

Table 2. Analytics certificates

Certificate name	Issuer	Description
`analytics-ai-endpoint or a7s-ai-endpoint`	`ingress-issuer`	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the `analytics-ai-endpoint` certificate. If this certificate is updated, restart the `mtls-gw` pod for the update to take effect.
`analytics-ca or a7s-ca`	`selfsigning-issuer`	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.
`analytics-client or a7s-client`	`analytics-ca`	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`analytics-server or a7s-server`	`analytics-ca`	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`<instance name>-storage`
		
		If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

Table 3. Portal certificates

Certificate name	Issuer	Description
`portal-ca or ptl-ca`	`selfsigning-issuer`	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.
`portal-client or ptl-client`	`portal-ca`	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.

Certificate name	Issuer	Description

`portal-server or ptl-server`	`portal-ca`	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
		
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`*.<instance name>-<site name>-db-all.<namespace>.svc`
		`*.<instance name>-<site name>-www-all.<namespace>.svc`
		`*.<instance name>-<site name>-db-all.<namespace>.svc.cluster.local`
		`*.<instance name>-<site name>-www-all.<namespace>.svc.cluster.local`
		`*.<namespace>.svc.cluster.local`
		`<instance name>-db`
		`#<remote portal CR name>-db # For 2DCDR only.`
		
		<instance name> and *<remote portal CR name>* are truncated if more than 15 characters.
		This certificate is used by all portal pods.

`portal-admin or ptl-director`	`ingress-issuer`	Server certificate used on the `portalAdminEndpoint`. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the `portal-admin-client` certificate. This certificate must have the same CA as the `portal-admin-client` certificate on the management subsystem. This certificate is used by the `portal-nginx` pod.
`portal-web`	`ingress-issuer`	The server certificate used on the `portalUIEndpoint`. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site. The default `portal-web` certificate is issued by the `ingress-issuer`, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate. This certificate is used by the `portal-web` ingress (`portal-web` route on OpenShift).
`ptl-replication-client`	`ingress-issuer`	2DCDR deployments only. Client certificate that is used in the warm-standby data center's `<remote-sitename>-www` and `<remote-sitename>-db` pods to connect to the active data center. This certificate is used by all the portal pods related to 2DCDR replication: `portal-db-remote`, `portal-www-remote`, and `portal-tunnel`.
`ptl-replication-server`	`ingress-issuer`	2DCDR deployments only. Server certificate used in the active data center's `<portal_CR>-tunnel` pod, the counterpart to the `ptl-replication-client` certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the `spec.multiSiteHA.replicationEndpoint` defined in the portal CR. This certificate is used by all the portal pods related to 2DCDR replication: `portal-db-remote`, `portal-www-remote`, and `portal-tunnel`.

Table 4. Gateway certificates

Certificate name	Issuer	Description
`gateway-peering or gw-peer`	`ingress-issuer`	This certificate secures the communication between gateways in your gateway cluster.
`gateway-service or gw-svc`	`ingress-issuer`	Legacy certificate that is not used.
`gwv6-endpoint or gw-gateway`	`ingress-issuer`	Legacy certificate that is not used.
`gwv6-manager-endpoint or gw-gateway-manager`	`ingress-issuer`	The `gatewayManager` endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the `gateway-client-client` certificate on management subsystem

Note: The API invocation certificate that callers to the published APIs on your gateways see is configured in the Cloud Manager UI when you register your gateway services. For more information about API invocation certificates, see Registering a gateway service and TLS profiles overview.

Table 5. Event gateway certificates -Cloud Pak for Integration only

Certificate name	Issuer	Description
`event-gateway-management-client`	`ingress-issuer`	The `event-gateway-management-client` certificate exists only on Cloud Pak for Integration deployments.

TLS certificates organized by usage

Table of API Connect certificates organized by where they are used.

Table 1. Management to gateway communication

Certificate name	Issuer	Description

Certificate name	Issuer	Description
`-ingress-ca`	`selfsigning-issuer`	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem `apim` pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the `portal-www` pods for the update to take effect.
`gateway-client-client or gw-dr-client`	`ingress-issuer`	Client certificate for communications with the gateway service. Restart the `apim`, and `taskmanager` pods after update. This certificate must have the same CA as the `gwv6-manager-endpoint or gw-gateway-manager` certificate on the gateway.
`gwv6-manager-endpoint or gw-gateway-manager`	`ingress-issuer`	The `gatewayManager` endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the `gateway-client-client` certificate on management subsystem

Table 2. Management to portal communication

Certificate name	Issuer	Description
`-ingress-ca`	`selfsigning-issuer`	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem `apim` pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the `portal-www` pods for the update to take effect.

`portal-admin-client or ptl-adm-client`	`ingress-issuer`	Client certificate used for communication with the portal subsystem on the `portalAdminEndpoint`. This certificate must have:
		
		• The same CA as the server certificate `portal-admin or ptl-director` on the portal subsystem.
		• The common name must match the `spec.adminClientSubjectDN` in the portal CR:
		
		`kubectl describe cert portal-admin-client`
		
		`Spec:`
		`  Common Name:  ptl-adm-client`
		
		`kubectl get ptl -o yaml`
		
		`  spec:`
		`    adminClientSubjectDN: CN=ptl-adm-client`
		
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=portal-admin-client`, or they could both be `CN=ptl-adm-client, O=cert-manager`, but they must be identical.

`portal-admin or ptl-director`	`ingress-issuer`	Server certificate used on the `portalAdminEndpoint`. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the `portal-admin-client` certificate. This certificate must have the same CA as the `portal-admin-client` certificate on the management subsystem. This certificate is used by the `portal-nginx` pod.

Table 3. Management and gateway to analytics communication

Certificate name	Issuer	Description

Certificate name	Issuer	Description
`-ingress-ca`	`selfsigning-issuer`	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem `apim` pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the `portal-www` pods for the update to take effect.

`analytics-ingestion-client or a7s-ing-client`	`ingress-issuer`	Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:
		
		• The same CA as the server certificate `analytics-ai-endpoint` on the analytics subsystem.
		• The common name must match the `spec.clientSubjectDN` in the analytics CR:
		
		`kubectl describe cert analytics-admin-client`
		
		`Spec:`
		`  Common Name:  a7s-ing-client`
		
		`kubectl get a7s -o yaml`
		
		`  spec:`
		`    clientSubjectDN: CN=a7s-ing-client`
		
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=a7s-ingestion-client`, or they could both be `CN=a7s-ingestion-client, O=cert-manager`, but they must be identical.
		
		To update this certificate, use kubectl. Restart the `apim`, `taskmanager`, and `analytics-proxy` pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.

`analytics-ai-endpoint or a7s-ai-endpoint`	`ingress-issuer`	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the `analytics-ai-endpoint` certificate. If this certificate is updated, restart the `mtls-gw` pod for the update to take effect.

Table 4. User-facing certificates

Certificate name	Issuer	Description
`api-endpoint or mgmt-platform-api`	`ingress-issuer`	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
`consumer-endpoint or mgmt-consumer-api`	`ingress-issuer`	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
`apim-endpoint or mgmt-api-manager`	`ingress-issuer`	API Manager UI server certificate.
`cm-endpoint or mgmt-admin`	`ingress-issuer`	Cloud Manager UI server certificate.
`portal-web`	`ingress-issuer`	The server certificate used on the `portalUIEndpoint`. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site. The default `portal-web` certificate is issued by the `ingress-issuer`, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate. This certificate is used by the `portal-web` ingress (`portal-web` route on OpenShift).
`hub-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `hub-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`turnstile-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `turnstile-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.

Table 5. Inter-subsystem certificates

Certificate name	Issuer	Description

Certificate name	Issuer	Description
`-ingress-ca`	`selfsigning-issuer`	The CA and issuer of all API Connect inter-subsystem and user-facing certificates. If there is a problem with this certificate, then all API Connect subsystems are inaccessible and unable to sync with each other. Update this certificate with **kubectl**. When updated, all child certificates must also be updated by deleting their corresponding secrets. See ingress-ca renewal. The ingress-ca certificate is also stored in the management subsystem database. It is visible from the Cloud Manager UI, in the truststores of the analytics, gateway, event gateway, and portal default TLS client profiles. When the management subsystem `apim` pod is restarted the certificate is reloaded into the database from the Kubernetes ingress-ca certificate. Do not attempt to update this certificate from the Cloud Manager UI. In 2DCDR deployments, and when you have subsystems in different namespaces, it is necessary to manually copy the ingress-ca certificate from the management subsystem to the other subsystems. Steps are provided in the 2DCDR and multi-namespace install sections of this documentation. If this certificate is updated, and you have a portal deployed, then restart the `portal-www` pods for the update to take effect.

`analytics-ingestion-client or a7s-ing-client`	`ingress-issuer`	Client certificate used for communication with the analytics subsystem on the ingestion endpoint. This certificate must have:
		
		• The same CA as the server certificate `analytics-ai-endpoint` on the analytics subsystem.
		• The common name must match the `spec.clientSubjectDN` in the analytics CR:
		
		`kubectl describe cert analytics-admin-client`
		
		`Spec:`
		`  Common Name:  a7s-ing-client`
		
		`kubectl get a7s -o yaml`
		
		`  spec:`
		`    clientSubjectDN: CN=a7s-ing-client`
		
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=a7s-ingestion-client`, or they could both be `CN=a7s-ingestion-client, O=cert-manager`, but they must be identical.
		
		To update this certificate, use kubectl. Restart the `apim`, `taskmanager`, and `analytics-proxy` pods after the update. This certificate is also used by the gateways for sending API events to the analytics subsystem. The updated certificate is sent by the management subsystem to the gateways, it is not necessary to restart any gateway pods.

`portal-admin-client or ptl-adm-client`	`ingress-issuer`	Client certificate used for communication with the portal subsystem on the `portalAdminEndpoint`. This certificate must have:
		
		• The same CA as the server certificate `portal-admin or ptl-director` on the portal subsystem.
		• The common name must match the `spec.adminClientSubjectDN` in the portal CR:
		
		`kubectl describe cert portal-admin-client`
		
		`Spec:`
		`  Common Name:  ptl-adm-client`
		
		`kubectl get ptl -o yaml`
		
		`  spec:`
		`    adminClientSubjectDN: CN=ptl-adm-client`
		
		For a two data center disaster recovery deployment, both data centers must have an identical subject name. For example, both data centers subject name could be `CN=portal-admin-client`, or they could both be `CN=ptl-adm-client, O=cert-manager`, but they must be identical.

`gateway-client-client or gw-dr-client`	`ingress-issuer`	Client certificate for communications with the gateway service. Restart the `apim`, and `taskmanager` pods after update. This certificate must have the same CA as the `gwv6-manager-endpoint or gw-gateway-manager` certificate on the gateway.
`api-endpoint or mgmt-platform-api`	`ingress-issuer`	Platform API endpoint server certificate. The certificate that is presented to callers of the platform REST API, and to the toolkit CLI.
`consumer-endpoint or mgmt-consumer-api`	`ingress-issuer`	Consumer API endpoint server certificate. The certificate presented that is presented to callers of the consumer REST API, and to the toolkit CLI.
`apim-endpoint or mgmt-api-manager`	`ingress-issuer`	API Manager UI server certificate.
`cm-endpoint or mgmt-admin`	`ingress-issuer`	Cloud Manager UI server certificate.
`mgmt-replication-client`	`ingress-issuer`	2DCDR deployments only. Client certificate used in the warm-standby data center's `<remote-sitename>-postgres` pod to connect to the active data center.
`mgmt-replication-server`	`ingress-issuer`	2DCDR deployments only. Server certificate used in the active data center's `<management_CR>-tunnel` pod. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the `spec.multiSiteHA.replicationEndpoint` defined in the management CR.
`hub-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `hub-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.
`turnstile-endpoint`	`ingress-issuer`	Used by the Automated API behavior testing application. User-facing server certificate for the `turnstile-endpoint`. If there is a problem with this certificate, then the user's browser shows a warning or error message.

Certificate name	Issuer	Description
`analytics-ai-endpoint or a7s-ai-endpoint`	`ingress-issuer`	Server certificate used on the analytics ingestion endpoint, in the mtls-gw pod. Management and gateway subsystems communicate with the analytics subsystem on this endpoint. The client certificates that are used by the management and gateway subsystems must use the same CA certificate as the **`analytics-ai-endpoint`** certificate. If this certificate is updated, restart the **`mtls-gw`** pod for the update to take effect.
`portal-admin or ptl-director`	`ingress-issuer`	Server certificate used on the **`portalAdminEndpoint`**. The management subsystem communicates with the portal subsystem on this endpoint. The corresponding client certificate used in the management to portal communication is the **`portal-admin-client`** certificate. This certificate must have the same CA as the **`portal-admin-client`** certificate on the management subsystem. This certificate is used by the **`portal-nginx`** pod.
`portal-web`	`ingress-issuer`	The server certificate used on the **`portalUIEndpoint`**. It is the server certificate that is presented to portal site users, and which their browser authenticates. If there is a problem with this certificate, then users see an error message about an invalid or insecure certificate in their browser when they access a portal site. The default **`portal-web`** certificate is issued by the **`ingress-issuer`**, which has a self-signed root certificate. Portal users might see a browser warning about use of a self-signed certificate. This certificate is used by the **`portal-web`** ingress (**`portal-web`** route on OpenShift).
`ptl-replication-client`	`ingress-issuer`	2DCDR deployments only. Client certificate that is used in the warm-standby data center's **`<remote-sitename>-www`** and **`<remote-sitename>-db`** pods to connect to the active data center. This certificate is used by all the portal pods related to 2DCDR replication: **`portal-db-remote`**, **`portal-www-remote`**, and **`portal-tunnel`**.
`ptl-replication-server`	`ingress-issuer`	2DCDR deployments only. Server certificate used in the active data center's **`<portal_CR>-tunnel`** pod, the counterpart to the **`ptl-replication-client`** certificate. This certificate must contain the DNS Subject Alternative Name of this data center's hostname and must be valid for the **`spec.multiSiteHA.replicationEndpoint`** defined in the portal CR. This certificate is used by all the portal pods related to 2DCDR replication: **`portal-db-remote`**, **`portal-www-remote`**, and **`portal-tunnel`**.
`gateway-peering or gw-peer`	`ingress-issuer`	This certificate secures the communication between gateways in your gateway cluster.
`gwv6-endpoint or gw-gateway`	`ingress-issuer`	Legacy certificate that is not used.
`gwv6-manager-endpoint or gw-gateway-manager`	`ingress-issuer`	The **`gatewayManager`** endpoint certificate. This is the server certificate on the gateway director endpoint, which the management subsystem communicates with. This certificate must be signed by the same CA as the **`gateway-client-client`** certificate on management subsystem
`event-gateway-management-client`	`ingress-issuer`	The **`event-gateway-management-client`** certificate exists only on Cloud Pak for Integration deployments.

Table 6. Intra-subsystem certificates

Certificate name	Issuer	Description
`management-ca or mgmt-ca`	`selfsigning-issuer`	The issuer for the management subsystems intra-subsystem certificates: management-client, management-server, postgres, and nats certificates. Communication between management subsystem pods fails if there is a problem with this certificate. This certificate is also used as the CA for REST API calls to the management subsystem from the other subsystems, when using **`in-cluster`** communication. See In-cluster service communication between subsystems
`management-client or mgmt-client`	`management-ca`	Client certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate.
`management-server or mgmt-server`	`management-ca`	Server certificate used in communication between management subsystem pods. Communication between management subsystem pods fails if there is a problem with this certificate. Required DNS names in the certificate: **`*.<namespace>`** **`*.<namespace>.svc`** **`*.<instance name>-server.<namespace>.svc`** **`<instance name>-server`**
`db-client-apicuser`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `db-client-pgbouncer`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`db-client-postgres`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `db-client-primaryuser`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`postgres`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `postgres-pgbouncer`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
v10.0.6 only `postgres-operator`	`management-ca`	Intra-subsystem certificate for the management database subsystem.
`natscluster-mgmt`	`management-ca`	Intra-subsystem certificate for the **`nats`** pods.

Certificate name	Issuer	Description
v10.0.6 only `db-client-replicator`	`management-ca`	2DCDR deployments only. Client certificate used by the `<management_CR>-tunnel` pod to connect to the other data center's `<management_CR>-tunnel` pod.
`analytics-ca or a7s-ca`	`selfsigning-issuer`	The issuer for the analytics-client and analytics-server certificates. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.
`analytics-client or a7s-client`	`analytics-ca`	Client certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate. If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`analytics-server or a7s-server`	`analytics-ca`	Server certificate used in communication between analytics subsystem pods. Communication between analytics subsystem pods fails if there is a problem with this certificate.
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`<instance name>-storage`
		
		If this certificate is updated, restart the `storage` and `ingestion` pods for the update to take effect.

`portal-ca or ptl-ca`	`selfsigning-issuer`	The issuer for the portal-client and portal-server certificates. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.
`portal-client or ptl-client`	`portal-ca`	Client certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate. This certificate is used by all portal pods.

`portal-server or ptl-server`	`portal-ca`	Server certificate used in communication between portal subsystem pods. Communication between portal subsystem pods fails if there is a problem with this certificate.
		
		Required DNS names in the certificate:
		
		`*.<namespace>`
		`*.<namespace>.svc`
		`*.<instance name>-server.<namespace>.svc`
		`<instance name>-server`
		`*.<instance name>-<site name>-db-all.<namespace>.svc`
		`*.<instance name>-<site name>-www-all.<namespace>.svc`
		`*.<instance name>-<site name>-db-all.<namespace>.svc.cluster.local`
		`*.<instance name>-<site name>-www-all.<namespace>.svc.cluster.local`
		`*.<namespace>.svc.cluster.local`
		`<instance name>-db`
		`#<remote portal CR name>-db # For 2DCDR only.`
		
		<instance name> and *<remote portal CR name>* are truncated if more than 15 characters.
		This certificate is used by all portal pods.

Troubleshooting API Connect

This section provides some general troubleshooting steps for use when encountering common problems with API Connect.

About

The following sections provide advice on diagnosing some common problems. This information can help you to identify whether a specific API Connect component is failing, whether it's an environmental problem, and whether you should raise a support request. Some advice includes reviewing specific log files so you can narrow down the source of the problem.

- Unable to access Management UIs
- Unable to access the portal sites
- Login failure on Management UIs
- Login failure on Portal sites
- Management operations not reflected on Gateway or Portal

Related tasks

- Gathering post-mortem logs
- Gathering logs for a VMware environment

Related information

- IBM Support Service Requests and PMRs

Unable to access Management UIs

If you are unable to access the UI login page, getting a blank page, an HTTP 404, or some other error.

When this happens it means that either the Management subsystem is down, or the access attempt is not reaching the Management subsystem due to a problem in the environment.

Note: If you are on VMware, some of the following checks involve running **kubectl** commands on your VMs. To access your VMs for running **kubectl** commands:

1. **SSH** to one of your subsystem VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. **Sudo** to the root user: `sudo -i`
3. From here you can run `kubectl` commands, for example:

```
# kubectl get pods
NAME                                                      READY   STATUS      RESTARTS      AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch           1/1     Running     1 (97d ago)   113d
...
```

4. API Connect pods run in the default namespace on VMware, so you do not need to specify the namespace with `-n` in the **kubectl** commands.

Check the following:

- Is the access attempt actually reaching the Management subsystem? To confirm this, check what is in the logs of the `juhu` pods which run on the Management subsystem. The `juhu` pods are the entry point for all UI, REST, and CLI requests to the Management subsystem. For every access request there should be line logged by the `juhu`pod, for example: When the user attempts to access the admin UI login page, the following is logged by the juhu pod:

  ```
  kubectl logs abc-management-juhu-fc8f6bb5d-jnw2q --since=5m
  ...
  9.20.85.205 - - [27/May/2022:10:35:49 +0000] [675fa59030a1a68028712ad2fdb2fdb1] "GET /api/cloud/admin/identity-providers
  HTTP/1.1" 200 301 "https://apimsvc20.hursley.ibm.com/auth/admin/sign-in/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" "9.145.13.216" [127.0.0.1:2000]
  request_time=0.222 upstream_response_time=0.224
  ...
  ```

  Note that the timestamp of these log records are not the time when the request arrived, but the time when the response to the request was sent back. To work out the arrival time you can subtract the `request_time=0.222` that appears later in the log message. If there is nothing logged in the `juhu` pods at the time the UI access is attempted, then check if you have any network problems between your browser and the Management subsystem. If you are on OVA then the tcpdump utility is available which you can use to confirm if traffic is reaching your Management VM, for example `tcpdump -i any src host 9.20.86.22 and dst port
  443`
- Are the Management subsystem pods all up and running? Check this with `kubectl get ManagementCluster -n <namespace>`. If this does not show all pods running, then check which pods are failing: `kubectl get pods -n <namespace>`. If you are on OVA then gather postmortem logs and open a support case at this point. Otherwise it's worth checking the reasons why the pods are failing to start with `kubectl describe` as it might be something in your kubernetes environment that you can fix. Note that on the Management subsystem the `postgres` pods will need to be running first, so if these are not running then most other Management pods will not start.
- Check the logs of the `apim` pods while attempting to access the UI. The `juhu` pods receive the requests from the UI, and send them on to the `apim` pods for processing. The logs of the `apim` pods may reveal some clues as to the cause of the problem. If nothing is logged in the `apim` pods at the time of the access attempt then it suggests a problem in the environment, the `juhu` pods unable to forward requests to `apim`.

Related tasks

- Gathering post-mortem logs
- Gathering logs for a VMware environment

Related information

- IBM Support Service Requests and PMRs

Unable to access the portal sites

If you are unable to access the login page of a portal site, getting a blank page, an HTTP 404, or some other error.

A portal site outage can occur for one of the following reasons:

- The portal site is down.
- The portal subsystem is down (and so all sites are down).
- The access attempt is not reaching the portal due to a problem in the network or Kubernetes environment.

Note: If you are on VMware, some of the following checks involve running **kubectl** commands on your VMs. To access your VMs for running **kubectl** commands:

1. **SSH** to one of your subsystem VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. **Sudo** to the root user: `sudo -i`
3. From here you can run `kubectl` commands, for example:

```
# kubectl get pods
NAME                                                      READY   STATUS      RESTARTS      AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch           1/1     Running     1 (97d ago)   113d
...
```

4. API Connect pods run in the default namespace on VMware, so you do not need to specify the namespace with `-n` in the **kubectl** commands.

Check for the following problems:

- Does the access attempt reach the portal subsystem?

To confirm that your browser is able to reach the portal site, check the `admin` container logs of the portal's `www` pods. The `nginx` process that runs in the `admin` container is the entry point for site access. For every attempt to access a portal site, the `admin` container logs the HTTP call, for example:

```
# kubectl logs abc-portal-site1-www-0 -c admin | grep nginx
...
[     nginx stdout]   584 ce6ce0:adad58:00580c 2022-05-27 12:57:38: "9.145.11.161, 9.20.85.206" portal.site1.example.com
"GET /providerorg1/sandbox/ HTTP/1.1" 200 5026 5315 5992 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36" 0.330 "-"
...
```

Note: On v10.0.6, v10.0.5.4 and previous releases, the `nginx` logs are in the `web` container. Use the same command, but replace "`-c admin`" with "`-c web`".
If the `admin` container logs nothing at the time the site access is attempted, then check for:

- Network problems between your browser and the portal subsystem.
- Problems with Kubernetes ingresses or OpenShift routes to the portal web endpoint.
- If you are on VMware, then the `tcpdump` utility is installed, and you can use it to confirm if traffic is reaching your portal VM. For example:

  ```
  tcpdump -i any src host 9.20.86.22 and dst port 443
  ```

- Are the portal subsystem pods all up and running?
  Run `kubectl get ptl` to check that your portal pods are running. If this does not show that all pods are running, then check which pods are failing: `kubectl get pods`. If you are on VMware, then gather postmortem logs and open a support case. If you are on Kubernetes or OpenShift, then check why the pods are failing to start with `kubectl describe <pod name>`. On the portal subsystem, the `-db-` pods must be running first; the other pods do not start until the `-db-` pods are running.

- If the `admin` container logs show that the access attempt is received, then check the `web` container logs to see whether there are any errors at the PHP level.

Related tasks

- Gathering post-mortem logs
- Gathering logs for a VMware environment

Related information

- IBM Support Service Requests and PMRs

Login failure on Management UIs

If you are able to access the Management UIs login page, but the login attempt is failing.

Note: If you are on VMware, some of the following checks involve running **kubectl** commands on your VMs. To access your VMs for running **kubectl** commands:

1. **SSH** to one of your subsystem VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. **Sudo** to the root user: `sudo -i`
3. From here you can run `kubectl` commands, for example:

   ```
   # kubectl get pods
   NAME                                                              READY   STATUS      RESTARTS      AGE
   abc-management-analytics-proxy-5f8bcd74c4-jbhch                   1/1     Running     1 (97d ago)   113d
   ...
   ```

4. API Connect pods run in the default namespace on VMware, so you do not need to specify the namespace with `-n` in the **kubectl** commands.

Check the logs of the `juhu` and `apim` pods at the time of the login attempt to confirm that the login attempt did reach the Management subsystem. For more information on the `juhu` logs see Unable to access Management UIs. If the login attempt reached the Management subsystem, more details of the login failure for Local User Registry users can be found in the `lur` pod logs, and for LDAP users in the `ldap` pods. Example:

```
# kubectl logs abc-management-lur-8bbc47b54-9vl8q --since=1m -f
...
2022-05-27T13:41:13.511Z audit [4d55be1800dcef18c64e0470484e6eac] START: >>>>>>>> [POST]: /user-login
2022-05-27T13:41:13.704Z lur:error [4d55be1800dcef18c64e0470484e6eac] Requested login, but user, notauser not found in user
registry,
2022-05-27T13:41:13.706Z bhendi:error [4d55be1800dcef18c64e0470484e6eac] Error in POST post:/user-login (user.js:login)
  - status : 401
...
```

Related tasks

- Gathering post-mortem logs
- Gathering logs for a VMware environment

Related information

- IBM Support Service Requests and PMRs

Login failure on Portal sites

If you are able to access the Portal sites login page, but the login attempt is failing.

Note: If you are on VMware, some of the following checks involve running **kubectl** commands on your VMs. To access your VMs for running **kubectl** commands:

1. **SSH** to one of your subsystem VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. **Sudo** to the root user: `sudo -i`
3. From here you can run `kubectl` commands, for example:

```
# kubectl get pods
NAME                                                              READY   STATUS      RESTARTS      AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch                   1/1     Running     1 (97d ago)   113d
...
```

4. API Connect pods run in the default namespace on VMware, so you do not need to specify the namespace with `-n` in the **kubectl** commands.

All Portal users except for the site Admin user are ultimately authenticated by the Management subsystem, however Portal adds additional security controls such as IP banning and Flood control which should be first place to check for portal login issues: Managing Developer Portal security.

If the login problem is with the Admin user, then check the logs of the `web` container in the `www` pod at the time of the attempt, for example:

```
# kubectl logs abc-portal-site1-www-0 -c web --since=1m
...
[   php-fpm stderr]   585 ce6ce0:adad58:c3852e 2022-05-27 14:16:49: WARNING: [pool www] child 1023 said into stderr: "WATCHDOG:
[NOTICE] [user] Login attempt failed for admin. | user: anonymous | uri:
https://provorg1.sandbox.portal.apimsvc21.hursley.ibm.com/provorg1/sandbox/user/login | referer:
https://portal.site1.example.com/provorg1/sandbox/user/login
...
```

If the Portal site itself is not rejecting the user login, then check the Management server logs during the login attempt: Login failure on Management UIs. It may also be worth checking the logs of the `admin` container in the `www` pod, which manages the Portal end of the communications with the Management subsystem.

Every account in the Developer Portal, including across different user registries for the same site, must have a unique email address, including the site Admin account. For example, if you configure three different user registries for a particular Developer Portal site, the email address *alice@acme.com* can be used to log in to the site from only one of the user registries. The default email address for the Admin account is the email address of the Catalog owner. It is not possible to create a user account (and associated Consumer organization) with the same email address as the Admin account (or that of the Catalog owner if their email address is different). Any attempts to create an account with the same email address results in the new account not functioning correctly, and returning the following error message when trying to log in: `A user already exists with this email address`.

Management operations not reflected on Gateway or Portal

Where publishes or deletes executed on the Management subsystem are not reflected on the gateway or portal within a few minutes. These problems are often caused by a network failure between Management and the other subsystems, or due to TLS handshaking failures.

Note: If you are on VMware, some of the following checks involve running **kubectl** commands on your VMs. To access your VMs for running **kubectl** commands:

1. **SSH** to one of your subsystem VMs as the user 'apicadm', for example: `ssh apicadm@mgmt1.example.com`
2. **Sudo** to the root user: `sudo -i`
3. From here you can run `kubectl` commands, for example:

```
# kubectl get pods
NAME                                                              READY   STATUS      RESTARTS      AGE
abc-management-analytics-proxy-5f8bcd74c4-jbhch                   1/1     Running     1 (97d ago)   113d
...
```

4. API Connect pods run in the default namespace on VMware, so you do not need to specify the namespace with `-n` in the **kubectl** commands.

For Gateway, the first place to check is the processing status page: Reviewing a gateway's processing status

For Portal and Gateway check the logs of the `taskmanager` pods at the time of the publish attempt. These pods send the publish operations to the gateway and portals via webhooks. If there is a network problem you may see socket timeout errors in the logs of these pods, or TLS errors if there is problem with your certificates. A common problem are load-balancers doing TLS termination instead of passthrough (where JWT security is not used, see Using JWT instead of mTLS on Kubernetes and OpenShift, Using JWT instead of mTLS on OVA). If the logs suggest that the webhook was sent successfully then check the gateway director (gwd) logs on the Gateway, and the `www` pod `admin` container logs on the portal, for example:

```
kubectl logs abc-portal-site1-www-0 -c admin
```

Taskmanager pods execute "send" tasks to publish operations to the gateways and portals. The taskmanager pods log messages about the progress of tasks, so there should be messages in a taskmanager pod for a send task soon after applying a change on the Management subsystem. A "send running in pod" message like the following example means the send task is running:

```
2023-05-05 14:02:37.270 management-taskmanager-f5d6fd77-dmnnt taskmanager apim:taskmanager:info:taskProcessor [0d30189c-db39-
4260-bb1f-a75a3ac5f1ba] task id: b3933261-5799-473d-9b83-9e29c245fde8 / kind: send running in pod: management-taskmanager-
f5d6fd77-dmnnt / containerId: cri-o://d5afe6d3960e333c1ea5d8f95b79564e769bc2ed5de9d8baebebd84b3f3ad4be 269ms after claimed
```

If the taskmanager logs an error containing "dispatching task failed for task"(logged when attempting to start a task) or an error containing "Stale claimed task" (logged 15 minutes later), API Connect might have a problem running tasks. A small number of these errors might be reported from time to time, and the system recovers automatically. But if the errors continue being reported, restart the natscluster pods (all together) and then restart the taskmanager pods to recover by running the following commands:

```
kubectl -n <namespace> delete po -l app.kubernetes.io/name=natscluster
kubectl -n <namespace> delete po -l app.kubernetes.io/name=taskmanager
```

When raising a support case, be sure to include logs from both the Management subsystem and the affected subsystems gateway/portal, stating when the publish attempt was made.

Related information

- IBM Support Service Requests and PMRs

Externally linked support topics

Topics that are linked to externally, covering maintenance operations, known issues, and error handling.

- **Management subsystem Postgres failover steps V10.0.6.0**
  V10.0.6 only: If your Postgres leader pod is not the original Postgres leader, complete the failover steps to restore the original pod as leader.
- **Management database changes V10.0.7.0**
  V10.0.7.0 supports only S3 backups for the management subsystem.
- **Restore blocked on 2DCDR deployments**
  V10.0.7.0 and later. Management database restore is blocked on two data center disaster recovery deployments.

Management subsystem Postgres failover steps V10.0.6.0

V10.0.6 only: If your Postgres leader pod is not the original Postgres leader, complete the failover steps to restore the original pod as leader.

Before you begin

This topic applies only to three replica deployments of API Connect.

Follow the steps in this topic if one or more of the following conditions are reported:

- If the **apicops version:pre-upgrade** command reports:

  ```
  Current postgres leader is not the original leader
  ```

- Openshift and Cloud Pak for Integration only: Your API Connect cluster CR reports `Error` status, and the output of:

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

  shows:

  ```
  Original PostgreSQL primary is running as replica, please perform failover to original primary before attempting upgrade.
  Refer - https://ibm.biz/BdPLWU
  ```

- When you attempt to upgrade the management subsystem by editing the management CR to change the version and license and you see this error:

  ```
  Original PostgreSQL primary is running as replica, please perform failover to original primary before attempting upgrade.
  Refer - https://ibm.biz/BdPLWU
  ```

About this task

The management subsystem uses Postgres as the database. A three replica API Connect deployment has a Postgres cluster that consists of 1 Postgres leader, and 2 Postgres replicas.
Note: In Kubernetes terminology, every pod in a cluster is called a "replica". In Postgres terminology, a Postgres cluster consists of "members", one of the members is designated as the leader, and the others are designated as "replicas". During normal operation, the Postgres cluster member that is the leader can change; one of the Postgres replicas can become the leader, and the original leader become a replica.
If the current Postgres cluster leader in your management deployment is not the original leader, then problems can occur during upgrade. To prevent upgrade problems, follow the steps in this topic to identify the original Postgres leader and assign it as the leader again.
Note: If you have an OVA deployment, run the steps that are documented in this topic from your management virtual appliance, as the root user: Login to your management virtual appliance with an SSH client and the username `apicadm`, then switch to the root user with `sudo -i`.

Procedure

1. Assign your management subsystem namespace to a new environment variable called `$NAMESPACE`

   ```
   export NAMESPACE=<management namespace>
   ```

   Note: On OVA deployments, set this to "default":

   ```
   export NAMESPACE=default
   ```

2. Identify the original name of your Postgres cluster, and assign it to a new environment variable `$PGCLUSTER_NAME`.

   ```
   export PGCLUSTER_NAME=$(kubectl -n $NAMESPACE get pgcluster | awk 'NR==2{print $1}')
   ```

   Confirm the variable is set correctly with:

   ```
   echo $PGCLUSTER_NAME
   ```

3. Get a list of the PVCs whose names start with `$PGCLUSTER_NAME` in your API Connect namespace.

   ```
   kubectl -n $NAMESPACE get pvc $PGCLUSTER_NAME $PGCLUSTER_NAME-wal
   ```

   ```
   NAME                          STATUS   VOLUME              CAPACITY   ACCESS MODES   STORAGECLASS     AGE
   ```

```
m1-33253de3-postgres       Bound    local-pv-2b61c0bd   234Gi      RWO            local-storage   6h42m
m1-33253de3-postgres-wal   Bound    local-pv-bd893695   234Gi      RWO            local-storage   6h42m
```

Note: If no PVCs are returned then check the **apicops** error or management CR warning, if it says "The original primary PVC is not attached to any pods in this namespace", then do not proceed with these steps. Instead, to re-create your Postgres cluster with a new leader:

a. Take a new management database backup Backup and restore.

b. Restore the management database from the new backup.

4. Get the names of all the Postgres pods.

```
kubectl get pods -n $NAMESPACE -l 'role in (master,replica)'
```

```
NAME                                          READY   STATUS    RESTARTS   AGE
m1-33253de3-postgres-dbd7966db-ltfhq          1/1     Running   0          61s
m1-33253de3-postgres-dxxx-c54776d87-wldgb     1/1     Running   0          26m
m1-33253de3-postgres-xwjf-8577b6c669-9chcp    1/1     Running   0          26m
```

5. Identify which pod was the original Postgres leader.
Describe each Postgres pod to see which one is using the two PVCs identified in step 3.

```
kubectl -n $NAMESPACE describe pod <pod-name> | grep 'ClaimName'
```

```
k describe pod m1-33253de3-postgres-dbd7966db-ltfhq | grep ClaimName
    ClaimName:  m1-33253de3-postgres
    ClaimName:  m1-33253de3-postgres-wal

k describe pod m1-33253de3-postgres-dxxx-c54776d87-wldgb | grep ClaimName
    ClaimName:  m1-33253de3-postgres-dxxx
    ClaimName:  m1-33253de3-postgres-dxxx-wal

k describe pod m1-33253de3-postgres-xwjf-8577b6c669-9chcp | grep ClaimName
    ClaimName:  m1-33253de3-postgres-xwjf
    ClaimName:  m1-33253de3-postgres-xwjf-wal
```

In this example, the highlighted pod **m1-33253de3-postgres-dbd7966db-ltfhq** is the pod that is using the PVCs identified in step 3.
Set this pod to the environment variable *$ORIGINAL_POSTGRES_LEADER*:

```
export ORIGINAL_POSTGRES_LEADER=<pod-name>
```

Note: If none of the Postgres pods are using the PVCs identified in step 3, then do not proceed with these steps. Instead, to re-create your Postgres cluster with a new leader:

a. Take a new management database backup Backup and restore.

b. Restore the management database from the new backup.

6. Run the **patronictl list** command in each Postgres pod and examine the output.

```
kubectl -n $NAMESPACE exec -it <postgres-pod> -- patronictl list
```

Example output for each Postgres pod:

```
Defaulted container "database" out of: database, set-libpq-certs (init)
+-----------------------------------------+-----------------+---------+---------+----+-----------+
| Member                                  | Host            | Role    | State   | TL | Lag in MB |
+ Cluster: m1-33253de3-postgres (7039951359768572098) ---------+---------+---------+----+-----------+
| m1-33253de3-postgres-dbd7966db-ltfhq    | 192.168.51.112  | Replica | running |  8 |         0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb  | 192.168.112.17  | Replica | running |  8 |         0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Leader  | running |  8 |           |
+-----------------------------------------+-----------------+---------+---------+----+-----------+
```

Each Postgres pod should show the same output, although the order of the records might differ. The names in the **Member** column should match the names of the Postgres pods. Check the following:

- If the original Postgres pod that you identified in step 5 has a **Lag in MB** of zero and a **TL** the same as the **Leader** pod, then continue to step 7.
- If the pod labeled **Leader** shows a **TL** that is less than the other pods, or a non-null **Lag in MB**, then proceed no further and open a support case.
- If the original Postgres pod that you identified in step 5 shows a lower **TL** than the other pods, or a nonzero value in **Lag in MB**, then reinitialize that pod:

  ```
  kubectl exec -it $ORIGINAL_POSTGRES_LEADER -n $NAMESPACE -- patronictl reinit <cluster name>
  $ORIGINAL_POSTGRES_LEADER
  ```

  where:

  - *<cluster name>* is the name of the Postgres cluster that is shown in the **patronictl list** output:

    ```
    + Cluster: <cluster name> ...
    ```

  Run **patronictl list** again to confirm that the **Lag in MB** is reducing. Do not proceed until it is zero. If it is not reducing, open a support case.

7. Make the pod that you identified in step 5 the **Leader**.
Run:

```
kubectl -n $NAMESPACE exec -it <postgres pod> -- patronictl failover
```

and select your $ORIGINAL_POSTGRES_LEADER as the candidate to failover to:

```
Candidate ['m1-33253de3-postgres-dbd7966db-ltfhq', 'm1-33253de3-postgres-dxxx-c54776d87-wldgb'] []: m1-33253de3-postgres-
dbd7966db-ltfhq
Current cluster topology
+-----------------------------------------+-----------------+---------+---------+----+-----------+
| Member                                  | Host            | Role    | State   | TL | Lag in MB |
+ Cluster: m1-33253de3-postgres (7039951359768572098) ---------+---------+---------+----+-----------+
| m1-33253de3-postgres-dbd7966db-ltfhq    | 192.168.51.112  | Replica | running |  8 |         0 |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb  | 192.168.112.17  | Replica | running |  8 |         0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Leader  | running |  8 |           |
+-----------------------------------------+-----------------+---------+---------+----+-----------+
Are you sure you want to failover cluster m1-33253de3-postgres, demoting current master m1-33253de3-postgres-xwjf-
8577b6c669-9chcp? [y/N]: y
```

```
2021-12-10 22:07:02.15154 Successfully failed over to "m1-33253de3-postgres-dbd7966db-ltfhq"
+-----------------------------------------+-----------------+---------+---------+----+-----------+
| Member                                  | Host            | Role    | State   | TL | Lag in MB |
+ Cluster: m1-33253de3-postgres (7039951359768572098) ---------+---------+---------+----+-----------+
| m1-33253de3-postgres-dbd7966db-ltfhq    | 192.168.51.112  | Leader  | running |  8 |           |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb  | 192.168.112.17  | Replica | running |  8 |         0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Replica | stopped |    |   unknown |
+-----------------------------------------+-----------------+---------+---------+----+-----------+
```

8. Run **patronictl list** to confirm that the pod that was identified in step 5 is the new leader, and that `TL` and `Lag in MB` are reported to be the same in each pod.

```
kubectl exec -it <postgres-pod> -n $NAMESPACE -- patronictl list
```

```
+-----------------------------------------+-----------------+---------+---------+----+-----------+
| Member                                  | Host            | Role    | State   | TL | Lag in MB |
+ Cluster: m1-33253de3-postgres (7039951359768572098) ---------+---------+---------+----+-----------+
| m1-33253de3-postgres-dbd7966db-ltfhq    | 192.168.51.112  | Leader  | running |  9 |           |
| m1-33253de3-postgres-dxxx-c54776d87-wldgb  | 192.168.112.17  | Replica | running |  9 |         0 |
| m1-33253de3-postgres-xwjf-8577b6c669-9chcp | 192.168.117.219 | Replica | running |  9 |         0 |
+-----------------------------------------+-----------------+---------+---------+----+-----------+
```

Management database changes V10.0.7.0

V10.0.7.0 supports only S3 backups for the management subsystem.

In V10.0.7.0, the Crunchy Postgres operator is replaced with a different operator called EDB. The Postgres database itself remains the same, but the Kubernetes pods and jobs that manage the Postgres database are new.

The change in Postgres operator results in new CR status outputs, and new backup requirements and settings, which are explained in this topic.

Management upgrade warning

When you upgrade the API Connect operator to V10.0.7.0, the management CR reports a `Warning` state until the management subsystem is also upgraded to V10.0.7.0. The reason for the warning is that due to the database operator change in V10.0.7.0 the management subsystem must not be left on V10.0.6.0 while the operator is on V10.0.7.0.

S3 backup settings automatically updated

If your management subsystem is already configured for S3 backups, then no extra pre-upgrade steps are required. The upgrade process converts the existing S3 backup settings for the new management database in V10.0.7.0.

When you upgrade to V10.0.7.0 with S3 backups configured, an extra backup path folder /edb is appended to your backup path. The /edb path is to distinguish new V10.0.7.0 backups, from your previous V10.0.6.0 backups.

Update backup settings before upgrade

V10.0.7.0 supports only backups to an S3 object-store for the management database. If your V10.0.6.0 deployment uses SFTP or local backups, then you must reconfigure your management backup settings to use an S3 object-store.

Upgrade to V10.0.7.0 is blocked if management database backups are not configured to use an S3 object-store. To update your backup settings, see the topic for your platform:

- Reconfigure management backup settings on Kubernetes.
- Reconfigure management backup settings on OpenShift.
- Reconfigure management backup settings on VMware.
  Ensure that you use the **apicup** command from your source version when you reconfigure the backup settings. Do not use the V10.0.7.0 **apicup** command to update backup settings on your V10.0.6.0 deployment.

If you do not require management database backups, you can enable an opt-out parameter to override the upgrade block:

- VMware:
  Set `accept-no-backups` to `true`

  ```
  apicup subsys set <subsystem name> accept-no-backups=true
  ```

- Kubernetes and OpenShift individual subsystem installations: Add the following annotation to your management CR:

  ```
  metadata:
    annotations:
      apiconnect-operator/backups-not-configured: "true"
  ```

- Cloud Pak for Integration and OpenShift top-level CR installations: Add the following annotation to your APIConnectCluster CR:

  ```
  metadata:
    annotations:
      apiconnect-operator/backups-not-configured: "true"
  ```

Restore blocked on 2DCDR deployments

V10.0.7.0 and later. Management database restore is blocked on two data center disaster recovery deployments.

From V10.0.7.0, you cannot restore the management database on an API Connect two data center disaster recovery deployment. You must remove the 2DCDR configuration before you can restore the management database. For more information, see the 2DCDR backup and restore documentation for your platform:

- 2DCDR backup and restore considerations on Kubernetes, OpenShift, and Cloud Pak for Integration.
- 2DCDR backup and restore considerations on VMware